True*Grid*® User's Manual

A Guide and a Reference

by

Robert Rainsberger

VOLUME 2:

Geometry, Assembly, Global Properties, and Output

Version 2.3.0

XYZ Scientific Applications, Inc.

March 29, 2006

Copyright © 2006 by XYZ Scientific Applications, Inc. All rights reserved.

True*Grid*,® the **True*Grid***® Manual, and related products of XYZ Scientific Applications, Inc. are copyrighted and distributed under license agreements. Under copyright laws, they may not be copied in whole or in part without prior written approval from XYZ Scientific Applications, Inc. The license agreements further restrict use and redistribution.

XYZ Scientific Applications, Inc. makes no warranty regarding its products or their use, and reserves the right to change its products without notice. This manual is for informational purposes only, and does not represent a commitment by XYZ Scientific Applications, Inc. XYZ Scientific Applications, Inc. accepts no responsibility or liability for any errors or inaccuracies in this document or any of its products.

True*Grid* ®is a registered trademark of XYZ Scientific Applications, Inc.

Silicon Graphics and IRIS are registered trademarks of Silicon Graphics, Inc.
Microsoft and MS-DOS are trademarks of Microsoft Corporation.
Unix is a registered trademark of AT&T.
Sun Microsystems is a registered trademark of Sun Microsystems, Inc.
Phar Lap is a registered trademark of its trademark holder.
ANSYS is a registered trademark of Swanson Analysis Systems, Inc.
PATRAN is a registered trademark of PDA Engineering.
FLUENT is a trademark of Fluent, Inc.
TASCflow is a trademark of Advanced Scientific Computing Ltd.
ViewPoint is a trademark of ViewPoint, Inc.

Some other product names appearing in this book may also be trademarks or registered trademarks of their trademark holders.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Table of Contents

Table of Contents 3

I. Geometry Commands 17
1. 2D Curves 18
ld initialize a 2D curve 18
apld append curve segments 19
lcc concentric arcs 20
ckl check curvature 21
gset set 2D Curves window 21
mazt intersection tolerance 22
lcd load curve definition 22
lcinfo information about load curves 25
ld3d2d convert 3D curves to 2D curves 25
ldinfo information about 2D curves 26
ldprnt print coordinates 27
lrl rays 27
lrot rotate 28
lsca scale 29
lscx scale first coordinate 29
lscz scale second coordinate 30
lt translate 30
flcd piece-wise trigonometric load curve 31
edgefile identify file used for edge data 33
rln import an edge file 2D curve 35
rlns import all edge file 2D curves 35
2. 2D Curve Segment Dictionary 36
lp2 polygonal line - pairs 36
lq polygonal line - lists 37
lpil intersect 2 curves 37
lpta tangent to a circle 38
ltas tangent to 2 circles 39
lep elliptic arc 41
lod normal offset 42
lnof normal averaging offset 43
lfil fillet 44
lap arc by center and end point 44
lar arc by radius and 2 points 45
ltp tangent arc by radius and end point 46

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

lpt arc by radius and end tangent line . . . 46
lat fillet by radius and end tangent line . . . 47
lad arc by center and angle . . . 48
lvc point in polar coordinates . . . 49
lstl translate . . . 49
ltbc list of radial coordinates . . . 50
ltbo changes to the list of radial coordinates . . . 51
lint interpolation . . . 52
csp2 cubic spline . . . 53
fws2 Fowler-Wilson cubic spline . . . 56
ctbc polar cubic spline . . . 60
ctbo Modify a Polar Cubic Spline . . . 61
ftbc Fowler-Wilson polar cubic spline . . . 61
ftbo modify Wilson-Fowler polar cubic spline . . . 63
rseg import from edge file . . . 64
3. 2D Curve Display Commands . . . 65
lcv display a load curve . . . 65
lv display all 2D curves . . . 65
lvi display list of 2D curves . . . 65
lvs display a sequence of 2D curves . . . 66
4. Create 3D Curves . . . 66
curd 3D curve definition . . . 68
igc create/append an IGES curve to a 3D curve . . . 71
sdedge create/append curve by surface edge . . . 71
lp3 create/append polygon of segments . . . 73
contour create/append a contour line to a 3D curve . . . 74
csp3 create/append a 3D cubic spline curve . . . 75
bsp3 create/append a B-Spline curve . . . 79
nrb3 create/append a NURBS curve . . . 80
ld2d3d create/append a 2D curve converted to 3D . . . 82
intcur create/append a 3D curve by interpolation . . . 84
lp3pt create/append a 3D curve by pairs of defined points . . . 85
3dfunc create/append a parameterized function curve . . . 86
projcur create/append curve projected onto a surface . . . 87
pscur create/append curve projected onto a surface and smoothed . . . 87
arc3 create/append arc of a circle . . . 89
cpcd create/append a copy of a previously defined 3D curve. . . . 91
cpcds create/append a copy of previously defined 3D curves . . . 92
twsurf create/append the curve at the intersection of two surfaces . . . 93
5. Display 3D Curve . . . 94
dcd display a 3D curve . . . 94

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

dcds display a set of 3D curves from a list . . . 94
dacd display all 3D curves . . . 94
acd add a 3D curve to the picture . . . 95
acds add a list of 3D curves to the picture . . . 95
rcd remove a 3D curve from the picture . . . 95
rcds remove a list of 3D curves from the picture . . . 95
racd remove all 3D curves from the picture . . . 95
lacd list all of the active 3D curves in the picture . . . 95
6. Print 3D Curves . . . 95
cdinfo print information about the 3D curves . . . 95
7. Delete 3D Curve . . . 96
rmseg remove last segment from 3D curve . . . 96
delcd delete a 3D curve from the True*Grid*® data base . . . 96
delcds delete a list of 3D curves from the True*Grid*® data base . . . 97
8. Surfaces . . . 97
accuracy set accuracy of surface projections . . . 99
chkfolds check for folds in surfaces . . . 99
delsd delete a surface . . . 99
delsds delete a set of surfaces . . . 100
fetol feature extraction from polygon surfaces . . . 100
getol relative tolerance to tessellate curves and surfaces . . . 101
lcsd list all composite surfaces with a specified surface . . . 101
mvpn modify a surface node of a polygon surface . . . 101
npll set the quality of the projection to surfaces . . . 102
project project a point to surface(s) . . . 103
pvpn place a surface node of a polygon surface . . . 104
sd surface definition . . . 105
smgap small surface gap tolerance . . . 110
trsd transform a surface definition . . . 110
sdinfo list surfaces . . . 112
vd define a volume . . . 112
9. Surface Dictionary . . . 113
blend3 blend three bounding 3D curves to form a patch . . . 113
blend4 blend four bounding 3D curves to form a patch . . . 114
bsps B-spline . . . 116
bstl read the standard binary STL file . . . 118
cn2p infinite cone, defined by two points . . . 118
cone infinite cone, defined by radius and angle . . . 120
cp infinite generalized cylinder (extruded or lofted curve) . . . 121
cr 2D curve revolved about an axis . . . 122
crule3d cylindrical surface between two 3D curves . . . 123

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

crx	rotate 2D curve about x-axis	124
cry	rotate 2D curve about y-axis	125
crz	rotate 2D curve about z-axis	126
csps	fit a cubic spline surface through 3D data	127
cy	infinite cylinder	132
cy3	cylinder from 3 points	133
cyr2	cylinder from 2 points and radius	134
cyr3	cylinder from 3 points and radius	135
er	ellipsoid (ellipse revolved about an axis)	136
face	face of the present part (Part Phase only)	137
faceset	convert a face set into a surface	138
function	surface by three algebraic expressions	139
hermite	precision 2nd order spline surface	140
igess	import a parametric IGES surface	143
igesp	import an IGES plane	145
intp	interpolate a surface between two surfaces	147
iplan	infinite plane defined by an implicit function	148
mesh	surface by tabular points	149
nrbs	NURBS	150
nurbs	import a NURBS surface	151
pipe	sweep a pipe shape along an arbitrary 3D curve	153
pl2	plane specified by two points	154
pl3	plane specified by three points	155
pl3o	plane specified by 3 points and an offset	156
plan	infinite plane	157
poly	convert a polygon set into a surface	158
pr	paraboloid (parabola revolved about an axis)	159
r3dc	3D curve revolved about an axis	160
rule2d	ruled surface between two 2D curves	161
rule3d	ruled surface between two 3D curves	162
sds	union of surfaces	163
sp	sphere	165
stl	read the standard ASCII STL file	166
swept	sweep 2D curves along a 2D curve	167
ts	torus	168
xcy	to transform an infinite x-axis cylinder	169
xyplan	transform an infinite xy-plane	170
ycy	to transform an infinite y-axis cylinder	171
yzplan	to transform an infinite yz-plane	172
zcy	to transform an infinite z-axis cylinder	173
zxplan	to transform an infinite zx-plane	174

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

10. Surface Display Commands 174
asd add a surface to the picture 175
asds add surfaces to the picture 175
ansd add neighboring surfaces to the picture 176
dasd display all surfaces in the picture 178
dsd display a surface in the picture 178
dsds display several surfaces in the picture 178
lasd list the surfaces in the picture 178
rasd remove all surfaces from the picture 179
rsd remove a surface from the picture 179
11. Importing Geometry from CAD Programs 179
iges render geometry in an IGES file 183
igesfile open an IGES file 184
igescd render a sequence of IGES curves 185
igeslbls use IGES surface labels to name surfaces 186
igespd render a sequence of IGES planes 187
igessd render a sequence of IGES surfaces 188
nurbsd render a sequence of IGES NURBS surfaces 189
saveiges save IGES binary data 191
trimming controls the trimmed surface algorithm 191
ltrim set the surface trimming work space 191
useiges use saved binary IGES data 192
vpsd import ViewPoint surfaces 193
wrsd write surface using the ViewPoint format 196
12. Displaying Geometry from CAD Programs 196
alv add a CAD level to the picture 197
dlv display a single CAD level in the picture 197
dlvs display several CAD levels in the picture 197
rlv remove a CAD level from the picture 198
agrp add a CAD group to the picture 198
dgrp display a single CAD group in the picture 198
dgrps display several CAD groups in the picture 199
rgrp remove a CAD group from the picture 199

II. Assembly Commands - Merge Phase 200
1. Merging Parts 201
mnl write the merged nodes list 203
pn place a node at a new location 204
2. Diagnostics Commands 205
ajnp find node near point 205

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

cenref restore reference center for moment calculations 205
centroid moments and inertia 205
elm highlight elements within a *measure* interval 205
elmoff turn off highlighting from the elm command 206
info mesh model summary 206
mass mass table 207
measure choose a way to measure mesh quality at every element 207
pmass active part mass 210
reference reference point for moments and inertia 210
size dimensions of the mesh 210
smags detect detached small groups of elements 210
tmass total mass 211
3. Graphics Commands 211
backplane toggles back plane removal 211
dpic dumps all picture parameters 211
rpic reads the picture parameters 212
4. Tokens in the Picture 212
mlabs multiple labels and conditions displayed 212
condition specify type of condition/constraint to be displayed 213
labels specify type of label to be displayed 246
5. Animation Commands 263
av animate views - linear interpolation 263
avc animate views - cosine interpolation 264
shv show a saved view 264
sv save view 264
6. Exploded Views 265
exp reactivate exploded views 265
expoff turn off exploded views 265
iniexp initialize explode offset to zero 266
mexp offset each subset of materials past the previous subset 266
pexp offset each subset of parts past the previous subset 266
sclexp explode scale factor 267
7. Material Commands 267
tmm specify the total mass of a material 267
buoy specify a buoyancy condition for a list of materials 267
am add material to the picture 268
ams add a list of materials to the picture 268
dam display all materials in the picture 269
dms display a set of materials in the picture 269
dm display one material in the picture 269
ram remove all materials from the picture 269

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

rm remove one material from the picture . . . 270
rms remove a list of materials from the picture . . . 270
8. Interface Commands . . . 270
iss save interface segments . . . 270
si select nodes for the slave side of a nodal sliding interface . . . 270
9. Springs, Dampers, and Point Masses . . . 271
npm creates a new node and assigns a point mass to it . . . 271
pm assigns a point mass to a node of the mesh . . . 272
pminfo table of point masses information . . . 273
spring create/modify a numbered spring . . . 273
delspd Delete a numbered spring/damper . . . 275
delspds Delete a list of numbered springs/dampers . . . 275
10. Element Commands . . . 275
bm create a string of beam elements . . . 276
bms change the section properties of a set of beams . . . 285
delem delete a set of elements . . . 287
etd specify the element types to be displayed in the graphics . . . 288
rbe rigid body elements . . . 288
11. Part Types . . . 291
linear specify following parts to use linear elements . . . 291
quadratic specify following parts to use quadratic elements . . . 291
partmode part command indices format . . . 291
12. Displacements, Velocities, and Accelerations . . . 293
acc Cartesian prescribed nodal boundary acceleration . . . 293
bv prescribed boundary surface velocities for NEKTON . . . 294
deform assigns deformations to beam elements . . . 294
dis initial nodal displacement . . . 294
fd displacement boundary condition . . . 295
fv prescribed velocities . . . 295
frb prescribed rotational boundary . . . 296
fvv prescribed variable velocities . . . 297
vacc Cartesian prescribed variable nodal boundary acceleration . . . 297
ve initial nodal velocity . . . 298
13. Force, Pressure, and Loads . . . 299
fa fixed nodal rotations . . . 299
fc concentrated nodal loads . . . 299
ffc concentrated nodal load with a follower force . . . 300
fmom follower nodal moment . . . 300
mom nodal moment about one global coordinate axis . . . 301
pr pressure load . . . 301
14. Boundary and Constraint conditions . . . 302

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

b nodal displacement and rotation constraints . . . 302
cfc boundary conditions for the CF3D output option . . . 303
fbc FLUENT boundary conditions . . . 304
il identifies an inlet for fluid flow. . . . 305
infol print nodal information with a specific load/condition . . . 305
jt assign nodes to a joint . . . 306
lb local nodal displacement and rotation constraints . . . 307
mpc shared nodal (multiple point) constraints for a nodal set . . . 308
nr non-reflecting boundary . . . 308
ol identifies a face of the mesh as an outlet for fluid flow . . . 308
rigid create a rigid body from a nodal set . . . 308
rml remove specific loads or conditions on a set of nodes . . . 309
rsl restore specific loads or conditions on a set of nodes . . . 310
spotweld interactive selection of spot welds . . . 311
spw create a single spot weld . . . 312
spwd spot weld property definitions . . . 313
spwf spot welds for LSDYNA material 100 . . . 314
sw select nodes that may impact a stone wall . . . 314
syf assign faces to a numbered symmetry plane with failure . . . 315
trp create tracer particles . . . 315
15. Radiation and Temperature Commands . . . 316
bf bulk fluid . . . 316
cv boundary convection . . . 316
cvt convection thermal loads . . . 317
fl prescribed boundary flux . . . 317
ft prescribed temperature . . . 318
rb prescribed radiation boundary condition . . . 318
re radiation enclosure . . . 319
te constant temperature for all nodes . . . 319
tepro nodal temperature profile . . . 319
tm initial temperature . . . 320
vvhg volumetric heat generation w/ functional amplitude . . . 320
16. Electric Conditions . . . 321
efl electric flux boundary condition . . . 321
mp constant magnetic potential . . . 321
v constant nodal electrostatic potential boundary condition . . . 321
17. Sets . . . 322
adnset add nodes to an ordered node set . . . 322
crvnset order a segment of an ordered node set (3D curve) . . . 324
eset add/remove elements to/from a set of elements . . . 325
fset add/remove faces to/from a set of faces . . . 326

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

infol information on nodal loads . . . 327
mvnset move a subset of nodes in an ordered node set . . . 328
nset add/remove nodes to/from a set of nodes . . . 330
onset order a segment of a nodal set . . . 333
pset create or modify a polygon set . . . 334
rml remove specific loads or conditions on a set of nodes . . . 335
rsl restore specific loads or conditions on a set of nodes . . . 337
rvnset remove a node subset . . . 339

III. Global Commands . . . 340
1. Materials . . . 341
delmats delete a material definition . . . 341
2. Parts . . . 341
block create a brick-shaped part . . . 342
cylinder create a cylindrical part . . . 345
readmesh read a file containing a mesh . . . 349
blude extrude a set of polygons . . . 354
meshscal scale up the mesh density for all parts . . . 356
beam initialize a beam part in Cartesian coordinates . . . 356
cbeam initialize a beam part in cylindrical coordinates . . . 358
ap add a part to the picture . . . 359
aps add a list of parts to the picture . . . 359
dap display all parts . . . 359
dp display one part in the picture . . . 359
dps display a set of parts in the picture . . . 360
pinfo part information . . . 360
rap remove all parts from the picture . . . 360
rp remove one part from the picture . . . 360
rps remove a set of parts from the picture . . . 361
3. Motion . . . 361
rotation global initial velocities as a rigid body rotation . . . 361
velocity global initial velocities as a rigid body translation . . . 362
4. Boundary Conditions and Constraints . . . 362
detp detonation points or lines . . . 362
jd joint definition . . . 363
jtinfo write information about joints . . . 365
lsys define a local coordinate system for the lb command . . . 365
lsysinfo list all of the local coordinate systems . . . 366
plane define a boundary plane . . . 367
plinfo write information about defined boundary planes . . . 368

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

5. Radiation and Temperatures 368
temp global default constant temperature 368
bfd bulk fluid definition 369
6. Springs, Dampers, and Point Masses 369
spd define the properties of a set of springs or dampers 370
spinfo write information about springs and dampers 372
7. Interfaces 372
bbinfo block boundary interface information 372
getbb retrieve a block boundary from a part file 373
inttr block boundary transition element interpolation factor 374
mbb master block boundary from point data 374
sid sliding interface definition 375
siinfo list sliding interface definitions 382
8. Elements 383
bsd global beam cross section definition 383
bsinfo write information about defined beam cross sections 405
bind Hughes-Liu beam user-defined integration points 406
lsbsd list the defined beam cross sections. 407
offset add offset to numbered entities in the output 407
sind shell user-defined integration rules 408
9. Sets 408
delset delete a set 408
esetc element set comment 409
esetinfo report the element set names and number of elements 409
fsetc face set comment 409
fsetinfo report the face set names and number of faces 410
nsetc attach a comment to a node set 410
nsetinfo report the node set names and number of nodes 410
10. Coordinate Transformations and Part Replication 410
lct define local coordinate transformations 412
gct define global coordinate transformations 414
lev define a set of transformations to replicate a set of parts 416
pslv begin replicating multiple parts 419
pplv end replicating multiple parts 420
csca scale all coordinates of all following parts 421
xsca scale all x-coordinates of all following parts 421
ysca scale all y-coordinates of all following parts 421
zsca scale all z-coordinates of all following parts 421
xoff translate all x-coordinates of all following parts 421
yoff translate all y-coordinates of all following parts 422
zoff translate all z-coordinates of all following parts 422

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

gexch permute the coordinates of all following parts . . . 422
exch permute the coordinates of all following parts . . . 423
nerl rule used to number the nodes and brick elements . . . 423
gmi material number increment for global replication . . . 424
lmi material number increment for local replication . . . 425
gsii sliding interface number increment for global replication . . . 425
lsii sliding interface number increment for local replication . . . 426
11. Control Statements . . . 426
if begin an if... elseif ... else ... endif . . . 426
elseif add an option to an if statement . . . 429
else final option in an if statement . . . 430
endif end an if statement . . . 431
endwhile end a while statement . . . 431
while begin a loop iteration . . . 431
12. Merging Nodes . . . 434
bnstol between node set tolerance . . . 436
merge switch to the merge (assembly) phase . . . 437
mns merge node sets . . . 437
st set tolerance and merge surface nodes . . . 438
stp set tolerance and merge surface nodes, with diagnostics . . . 439
t set tolerance and merge nodes . . . 439
tp set tolerance and merge nodes, with diagnostics . . . 440
ztol minimum non-zero absolute coordinate . . . 440
bptol between parts tolerance specification . . . 440
ptol part tolerance specification . . . 441
rigbm identify two rigid bodies to be merged . . . 441
13. Global Graphics Commands . . . 441
ibzone control the computational window frame . . . 442
noplot turn all graphics off . . . 442
plot turn graphics back on . . . 442
14. Miscellaneous Commands . . . 443
becho echo and beep . . . 443
bulc locate butterfly triple point . . . 443
c beginning of a comment . . . 444
circent center of a circle . . . 444
crprod cross product . . . 445
curtyp default attach command . . . 445
dc desk calculator . . . 447
distance distance between two points . . . 447
echo echo a string . . . 448
end terminate True*Grid*® with no more output . . . 448

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

errmod set error handler mode . . . 448
expressions FORTRAN-like expressions . . . 449
def define a function . . . 451
include execute commands from batch file . . . 451
inprod inner or dot product . . . 452
interrupt switch from batch commands to interactive mode . . . 453
intyp set default mesh interpolation . . . 454
painfo print information about parameters . . . 454
para define parameters . . . 455
resume resume executing batch commands . . . 456
title assign title to the problem . . . 456
tpara typed parameters . . . 457
tricent find optimal center of a triangular structure . . . 457
subang angle between two intersecting lines . . . 458
caption assign a caption to the physical window . . . 459
mxp change number of mesh convergence passes . . . 459
trapt transform a point . . . 460

IV. Output . . . 463
1. Output File Format Commands . . . 464
comment pass a comment to the DYNA3D output file . . . 464
epb element print block for DYNA3D and LSDYNA . . . 465
mof mesh output file name . . . 465
ndigits number of digits written for coordinates . . . 466
npb node print block for DYNA3D and LSDYNA . . . 466
save dump buffered data to the tsave file . . . 466
verbatim write verbatim to the output file . . . 467
abaqus ABAQUS output format . . . 467
ale3d ALE3D output format . . . 467
ansys ANSYS output format . . . 467
autodyn AUTODYN-3D output format . . . 467
cf3d Convective Flow output format . . . 468
cfd-ace CFD-ACE output format . . . 468
cfx CFX output format . . . 468
dyna3d DYNA3D output format . . . 468
es3d ES3D output format . . . 468
exodusii Exodus II output format . . . 468
fidap FIDAP output format . . . 468
fluent FLUENT output format . . . 468
gemini GEMINI output format . . . 469

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

gridgen3d GRIDGEN output option . . . 469
iri IRI output format . . . 469
lsdyna LS-DYNA output format . . . 469
lsnike3d LS-NIKE3D output format . . . 469
marc MARC output format . . . 469
nastran NASTRAN output format . . . 469
nike3d NIKE3D output format . . . 470
nekton2d NEKTON2D output format . . . 470
nekton3d NEKTON3D output format . . . 470
ne/nastran NE/NASTRAN output format . . . 470
neutral Neutral output Format . . . 470
refleqs REFLEQS output format . . . 470
starcd STARCD output format . . . 470
plot3d PLOT3D output format . . . 471
poly3d Generic output format . . . 471
tascflow TASCflow output format . . . 471
topaz3d TOPAZ3D output format . . . 471
topaz3d2 TOPAZ3D version 2000 output format . . . 471
viewpoint VIEWPOINT output format . . . 471
2. Analysis Options . . . 472
abaqstep ABAQUS analysis step . . . 472
ansyopts ANSYS analysis option . . . 472
dynaopts DYNA3D analysis options . . . 472
lsdyopts LS-DYNA analysis and database options . . . 472
marcopts MARC analysis options . . . 472
nastopts NASTRAN analysis options . . . 472
nenstopt NE/NASTRAN analysis options . . . 472
nekopts 2d and 3d NEKTON 2.85 analysis options . . . 472
nikeopts NIKE3D analysis options . . . 472
lsnkopts LS-NIKE3D analysis options . . . 472
tz3dopts TOPAZ3D analysis options . . . 472
3. Material Definitions . . . 473
abaqmats ABAQUS materials . . . 473
ansymats ANSYS materials . . . 473
dynaeos DYNA3D equation of state . . . 473
dynamats DYNA3D materials . . . 473
fluemats FLUENT materials . . . 473
nastmats NASTRAN materials . . . 473
nenstmats NE/NASTRAN materials . . . 473
lsdymats LS-DYNA materials . . . 473
lsdythmt LS-DYNA thermal materials . . . 474

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

lsdyeos LS-DYNA3D equation of state 474
nikemats NIKE3D materials 474
lsnkmats LS-NIKE3D materials 474
marcmats MARC materials 474
patsmats PATRAN materials 474
tz3dmats TOPAZ3D materials 474

V. Appendix 475
Cartesian coordinate system 476
Cylindrical coordinate system 477
Spherical coordinate system 478

VI. Index 480

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

I. Geometry Commands

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

1. 2D Curves

These commands define and modify 2D curves. All 2D curves use a 2 dimensional local coordinate system where **x′** is the abscissa (horizontal axis) and **z′** is the ordinate. A 2D curve can be rotated around the **z′**-axis of the curve's local coordinate system to form a surface of rotation. You can choose how to orient this local **z′**-axis in the global coordinate system (see the **sd** command with the option **crx**, **cry**, **crz**, or **cr)**. A 2D curve can be extruded or lofted in the third direction to make an infinite surface (see the **sd** command using the **cp** option). Two 2D curves can be placed into the global 3D coordinate system and a ruled surface formed between them (see the **sd** command using the **rule2d** option). 2D curves can form the cross sections of a surface (see the **sd** command using the **swept** option). For more details, see the **sd** command, and the surface dictionary. You can embed a 2D curve in the global 3D coordinate system or turn the 2D curve into a 3D curve (see the **curd** command using the **ld2d3d** option).

Some 2D curves are special purpose and cannot be used to form 3D geometry. For example, load curves are used, when defining boundary conditions, to regulate a time dependent variable.

ld initialize a 2D curve

ld *2D_curve_# segments*

where a

2D_curve_#	number of the curve to be initialized
segment	2D curve segment from the 2D Curve Segment Dictionary

Remarks

You may use as many segments as you like; each is appended to the end of the previous one until the next 2D curve is initialized.

Using this command you construct a curve by appending curve segments to each other, end-to-end. The **ld** command initializes a new 2D curve definition. All subsequent 2D curve segments append to the end of this 2D curve. Once a new 2D curve is created, the previous 2D curve can no longer be appended. Many kinds of curve segments are available, and you may combine them in almost any order. For details, see the 2D Curve Segment Dictionary. This method of constructing 2D curves is fashioned after the way a draftsman might draw a complex 2D curve.

These 2D curves can be used to form 3D curves and surfaces. For example, a 2D curve can be rotated about any axis of symmetry, extruded indefinitely in any direction, or combined with another 2D curve to produce a ruled surface between the two 2D curves. See the **sd** command.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
ld 1 lp2 10.7 -2 11.4 -2;
ld 2 lp2 1 0;
lfil 0 10.7 -2 90 1;
lp2 10.7 -2;
ld 3 lp2 1 0 0 2.7 3.05 5.75;
ltp 3.75 7.5 2.55;
lp2 3.75 8.75;ltp 2.8 9.9 1.45;
lp2 0 11.2 0 14 8 13
  8.35 12.75;
ld 4 lp2 11.4 -2;
lfil 90 7.3 .6 0 1;
lfil 180 8.4 2.3 90 1.45;
lp2 8.4 2.3;
ld 5 lep .5 .35 6.3525 5.4475
  -180 -90 45;
lep .5 .8 8.75 2.65 90 180 45;
ld 6 lp2 6 5.09393 5.3 5.8;
ltp 4.65 7.5 2.6;
lp2 4.65 8.7;ltp 6.9 9.95 1.5;
lp2 7.45 9.6;ltp 8.8 9.4 2.5;
ltp 9.45 10.05 .65;
lp2 9.45 10.6 9.3 11.05 9.2
  11.05 9.05 10.6 9.05 10.2;
ltp 8.7 10.15 .25;lp2 7.55 10.85;
ltp 7.55 12.15 .8;lp2 8.35 12.75;
```

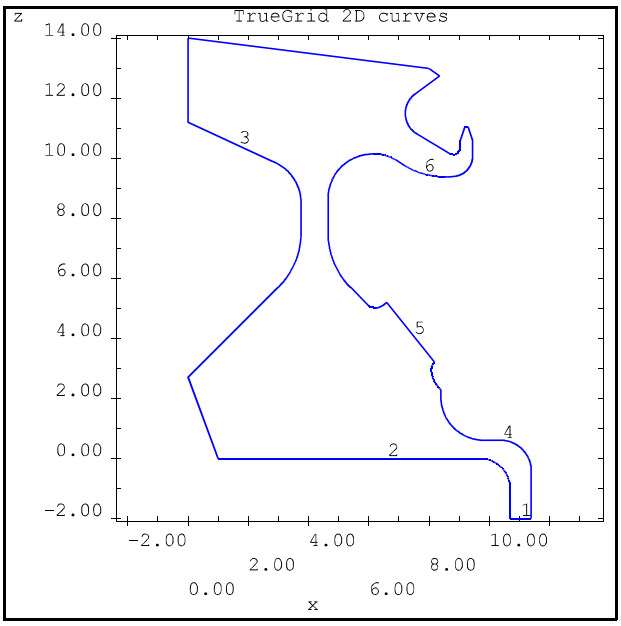


Figure 1 Complex 2D Geometry

apld append curve segments

apld *segments*

where a *segment* any one of the curve segments described in the 2D Curve Segments Dictionary

Remarks

A 2D curve is initialized with an **ld** command with options. Then this command can be used to append other 2D curve segments to the end of it. This command is used primarily by the GUI (graphical user interface) so that you have a command to click on in the menus when you want to interactively append another 2D curve segment to the previously initiated 2D curve. Otherwise, the **apld** command does nothing.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Examples

The following two curve definitions are equivalent:

```
ld 1 lp2 1.6 5 1.6 4;
lat 1.6 4 5 4 .2;
lat 2.8 4 3 5 .2;
```

and

```
ld 1 lp2 1.6 5 1.6 4;
apld lat 1.6 4 5 4 .2;
apld lat 2.8 4 3 5 .2;
```

lcc concentric arcs

lcc x' z' θ_{begin} θ_{end} $radius_1$ $radius_2$ ... $radius_n$;

where

(x', z')	coordinates of the center of the circle
θ_{begin}	angle which defines the beginning of the arc
θ_{end}	angle which defines the end of the arc
$radius_1$ . . . $radius_n$	radii of the various arcs

Remarks

The 2D curves are numbered sequentially, beginning with the last-defined 2D curve number, plus one.

Example

```
lcc 1 2 45 135 1 4 9 16 25
36;
```

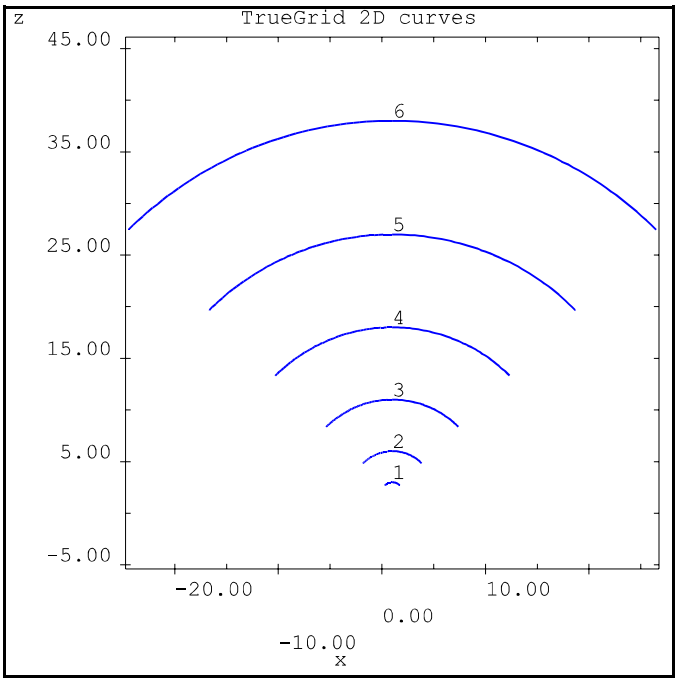


Figure 2 Concentric Arcs

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

ckl check curvature

ckl *2D_curve$_1$ 2D_curve$_2$ angle*

where

2D_curve$_1$	number of the first curve to be checked
2D_curve$_2$	number of the last 2D curve to be checked
angle	is the greatest permissible angular deviation from 180 degrees.

Remarks

This command will check each defined 2D curve whose number lies in the specified range. The angle at each point of the polygonal line approximation to the curve is calculated and. reported when this angle differs from 180 degrees by more than the specified angular deviation, *angle*. Then the specified curves are displayed..

Example

```
ld 1 lp2 0 0 1 1 2 1 2 0;
ckl 1 1 60
```

The above example will cause the following warning message:

warning - curve 1 at node 3 has an angle of 90.00 degrees

gset set 2D Curves window

gset *xc´ zc´ size*

where

(*xc´, zc´*)	coordinates of the center of the 2D Curves window
size	length of the square 2D Curves window

Remarks

The default center and size of the 2D Curves window is based on the 2D curves being drawn. This window is made just large enough to contain the selected curves for display while remaining a square. In order to return to this default, use a size of 0.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

mazt intersection tolerance

mazt *tolerance*

Remarks

The **mazt** command controls the tolerance for the intersection of the 2D curves. The default is .001. This tolerance is used in the **lpil** 2D curve segment type.

lcd load curve definition

lcd *ld_curve_# options curve ;*

where

ld_curve_#	load curve number

where an *option* can be

sidr *type*	
where *type* can be:	
0	transient analysis or other
1	stress initialization only
2	stress initialization and transient analysis
sfa *factor*	time (abscissa) scale factor
sfo *factor*	amplitude (ordinate) scale factor
offa *offset*	time (abscissa) offset
offo *offset*	amplitude (ordinate) offset
dattyp *type*	
where *type* can be	
0	monotonic time (abscissa)
1	non-monotonic time (abscissa)
g	modal damping viscous dampers
crit	modal critical damping
q	modal damping quality factor
d1	frequency/time dependency dynamic load
d2 *offset*	frequency/time dependency dynamic load
d3 *offset scale*	frequency/time dependency dynamic load
d4 *offset scale min*	frequency/time dependency dynamic load
m1	temperature dependence
m2 *offset*	temperature dependence
m3 *offset scale*	temperature dependence
m4 *offset scale min*	temperature dependence

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

where

offset	offset for amplitude
scale	scale factor for amplitude
min	lower clamp for the amplitude

s1 stress dependent

st material property temperature dependent

rnd1 power spectral density

where a *curve* must be one of

$t_1 f_1\ t_2 f_2 \ldots t_n f_n$

lp $t_1 f_1\ t_2 f_2 \ldots t_n f_n$

where (t_i, f_i) are the pairs of time and load amplitudes

func *npnts* $u_0\ u_x$ *load_expression*

where

npnts	number of points in curve
u_0	minimum value of the independent variable u
u_x	maximum value of the independent variable u
load_expression	algebraic expression with independent variable u

Remarks

The load curve number is used for identification and is unique for each load curve.

Load curves are usually for dynamic simulations, variable properties, or tables. They are used differently by different simulation codes. Sometimes a load command, such as **fc**, will require a load curve for completeness. Depending on your output selection, the load curve number you select in these types of commands will be used as a set id and will not require that you define a corresponding load curve with **lcd**.

The easiest way to make a load curve is:

```
lcd 1 0 0 .001 1;
```

which defines load curve 1 with two points. At time 0 the amplitude of the associated load will be 0. Then the amplitude is ramped up linearly to full amplitude at time .001. This is equivalent to:

```
lcd 1 lp 0 0 .001 1;
```

for backward compatibility.

The options **sidr**, **sfa**, **sfo**, **offa**, **offo**, and **dettyp** are for LS-DYNA load curves.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

The options **s1**, **st**, **rnd1**, **g**, **crit**, **q**, **d1**, **d2**, **d3**, **d3**, **m1**, **m2**, **m3**, and **m4** are for NASTRAN and NE/NASTRAN tables.

Example

```
lcd 3 0 0 1 1 2 1 2.00001 0 3 0;
```

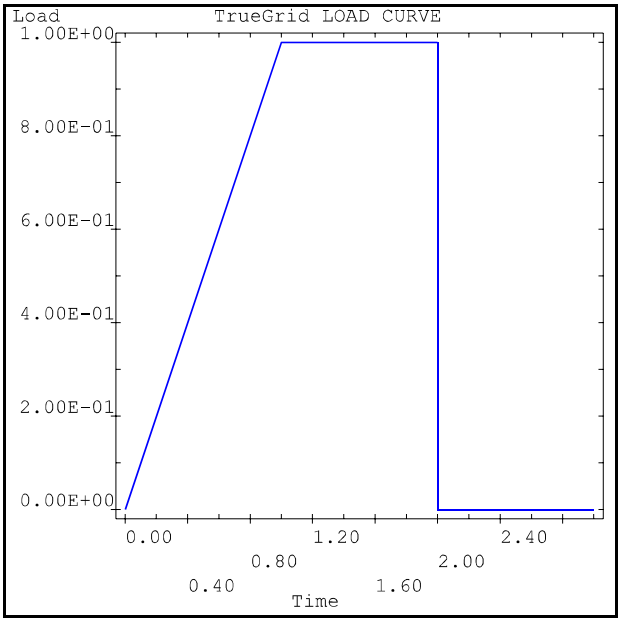


Figure 3 Load Curve Drawn Using **LCV**

Example

```
lcd 2 sidr 0 func 100 0 .001
sin(u*180000);
```

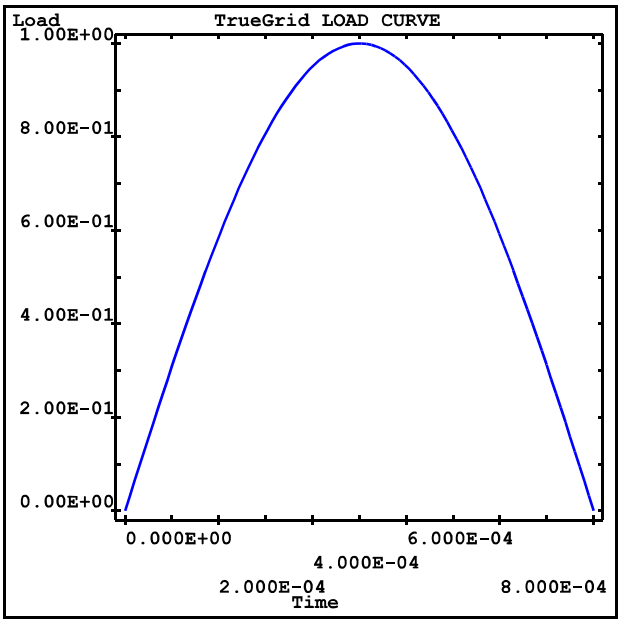


Figure 4 Function load curve

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

lcinfo information about load curves

lcinfo *<no arguments>*

Remarks

Prints the number of points of each load curve.

Example

```
flcd 1 phase 180 cosine 30 5 .1 lp .1 1; tinit .1 finit 1
fsca .1 foff 1 phase 0 sine 60 60 .1;
lcd 2 0 0 .1 .04 .4 .08 .9 .12 1.6 .16 2.5 .2;
lcinfo
```

The above example produces 2 load curves. The table below is written to the tsave file and to the text window if the command is issued interactively.

```
  LOAD CURVE DEFINITIONS
load curve number      1 has       91 points
load curve number      2 has        6 points
```

ld3d2d convert 3D curves to 2D curves

ld3d2d *reduct_coord first_2D_curve_# 3D_curve$_1$ 3D_curve$_2$ ... ;*

where

reduct_coord	**x**, **y**, or **z** coordinate by which to reduce the 3D curve to a 2D curve
first_2D_curve_#	is the number to be assigned to the first 2D curve
3D_curve$_i$	are the numbers of the 3D curves

Remarks

The symbols **x**, **y**, or **z** specify the coordinate to be eliminated. The specified 3D curve is projected onto the coordinate plane defined by the remaining two coordinates.

Example

One of the edges of a surface is converted into a 3D curve. The **ld3d2d** command converts the 3D curve into a 2D curve by eliminating the z-coordinate.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
sd 1 function 1000 2000 100 2000 u*(cos(v)+v*sin(v));
u*(sin(v)-v*cos(v)); 1000*(u+v*.25);;
curd 1 sdedge 1.2
ld3d2d z 1 1;
```

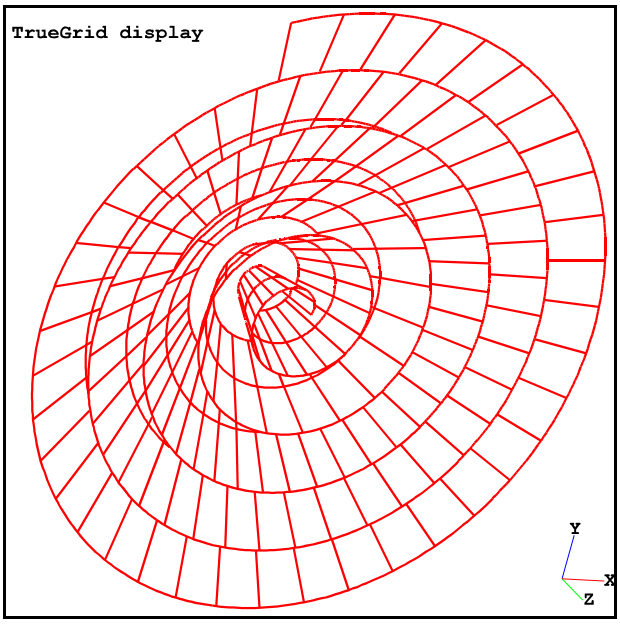


Figure 5 Spiral Surface

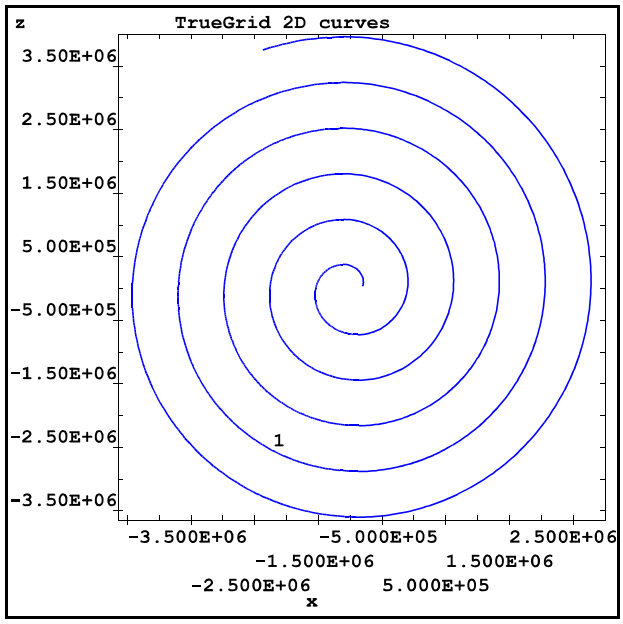


Figure 6 Curve From Spiral Surface

ldinfo information about 2D curves

ldinfo *<no arguments>*

Remarks

A table is printed listing each defined 2D curve. This data is always written to the tsave file. When in the interactive mode, the data is also written to the text window. This table contains a list of 2D curves that have been defined and the number of points used in their definition.

Example

For a model with several curves, this command was issued:

```
ldinfo
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
2D CURVE DEFINITIONS
  2D curve definition number   1 contains     73 points
  2D curve definition number   2 contains      8 points
  2D curve definition number   4 contains      3 points
  2D curve definition number   9 contains     13 points
  2D curve definition number  14 contains     16 points
  2D curve definition number  30 contains      4 points
last 2D curve defined is number     14
```

ldprnt print coordinates

ldprnt *2D_curve*

where *2D_curve* is the number of a defined 2D curve.

Remarks

The **x′** and **z′**-coordinates forming the polygonal approximation to a 2D curve are printed to the tsave file, and, when the command is issued interactively, to the text window.

Example

```
ldprnt 30

2D Curve Definition       30
     2.812500E+00   3.843750E+00
     2.812500E+00   3.875000E+00
     2.937500E+00   3.875000E+00
     2.937500E+00   3.843750E+00
```

lrl rays

lrl $x'_{center}\ z'_{center}\ length\ angle_1\ angle_2 \ldots angle_n$;

where

$(x'_{center}, z'_{center})$	is the point that the rays emanate from,
length	is the length of each ray, and
$angle_i$	is the angle formed by the ***i***-th ray and the **x′**-axis.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

The 2D rays are numbered sequentially, beginning with one more than the number of the last 2D curve that was defined.

Example

```
lrl 1 2 3 0 5 15 30 50 70 85 95
100;
```

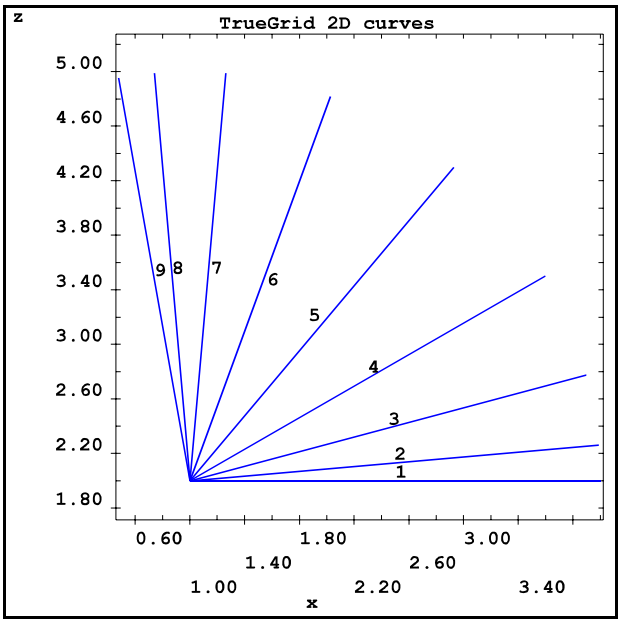


Figure 7 Radial Lines

lrot rotate

lrot *2D_curve angle*

where

2D_curve is the number of the 2D curve and

angle is the amount of rotation in degrees.

Remarks

This function rotates an existing 2D curve where a positive angle is counter-clockwise. See the following page for an example.

Example

```
ld 1 lp2 1 0;lap 3 -.1 2 2;lp2 5
-.1 5 .1 3 .1;
lap 1 .1 2 2.1;lp2 1 0;
ld 2 lp2 1 0;lap 3 -.1 2 2;lp2 5
-.1 5 .1 3 .1;
lap 1 .1 2 2.1;lp2 1 0;
lrot 2 45
```

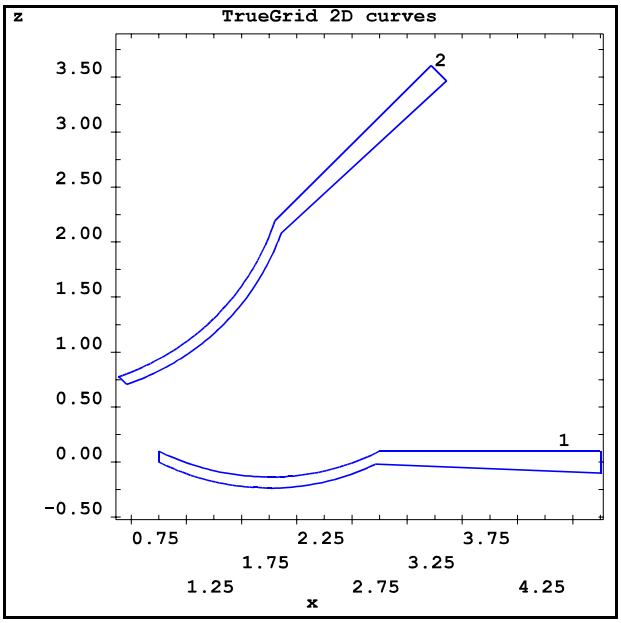


Figure 8 Rotated 2D curve

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

lsca scale

lsca *2D_curve scale*

where

2D_curve number of the 2D curve

scale scale factor

Remarks

Each coordinate of the curve are multiplied by the value of *scale*.

Example

```
ld 1 lp2 1.5 0;lfil 0 2 1 -75
.2;lfil -75 [cos(30)] [sin(30)] 30
.2 lfil 30 [cos(15)] [sin(15)] 105
.2 ;lfil 105 1.5 0 0 .2;lp2 1.5 0;
ld 2 lstl 1 1.525 .385;lsca 2 .5
```

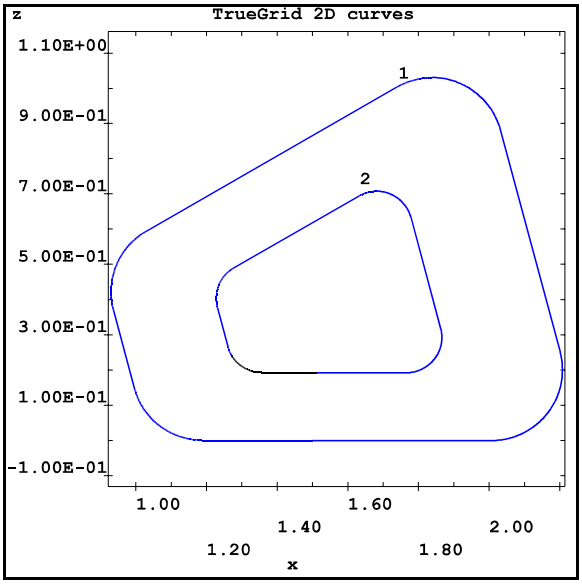


Figure 9 Scale Both Coordinates

lscx scale first coordinate

lscx 2D_curve *x´_scale*

where

2D_curve number of the 2D curve, and

x´_scale scale factor

Remarks

Each first coordinate on the curve is multiplied by *x´_scale*.

Example

```
ld 1 lp2 0 0;lad 1 0 360 ;
ld 2 lp2 0 0;lad 1 0 360;lscx 2 3
```

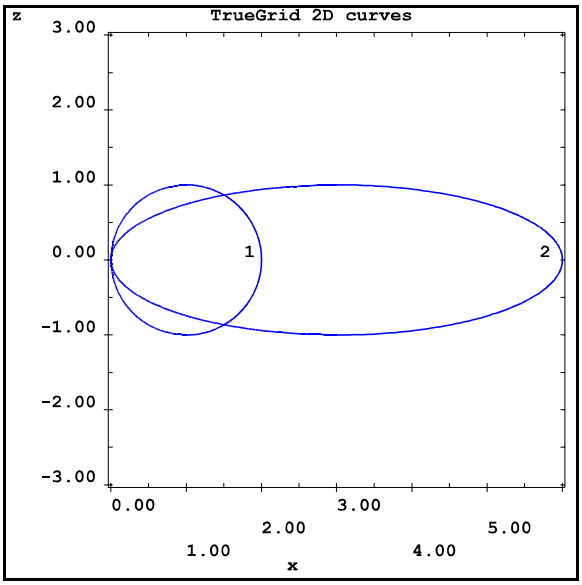


Figure 10 Scale First Coordinate Only

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

lscz scale second coordinate

lscz *2D_curve z′_scale*

where

2D_curve number of the 2D curve

z′_scale scale factor.

Remarks

Each second coordinate on the curve is multiplied by *z′_scale*.

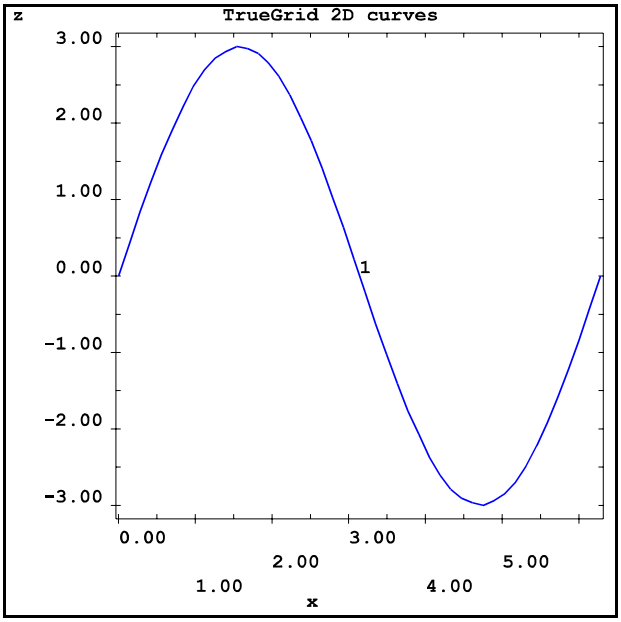


Figure 11 Scale Second Coordinate Only

Example

```
ld 1 lp2 0 0 .14 .14 .28 .28 .42
.41 .56 .53 .7 .64 .84 .74
.98 .83 1.12 .9 1.26 .95 1.4 .98
1.54  1  1.68  .99  1.82  .97
1.95.93 2.09 .87 2.23 .79 2.37
.69 2.51 .59 2.65 .47  2.79 .34
2.93 .21 3.07 .07 3.21 -.07 3.35
-.21 3.49  -.34 3.63 -.47 3.77
-.59 3.91 -.69 4.05 -.79 4.19
-.87 4.33 -.93 4.47 -.97 4.61
-.99 4.75 -1 4.89 -.98 5.03 -.95
5.17  -.9  5.31  -.83  5.45  -.74
5.59 -.64 5.72 -.53 5.86 -.41 6
-.28 6.14 -.14 6.28 0;lscz 1 3
```

lt translate

lt *2D_curve Δx′ Δz′*

where

2D_curve is the number of a previously defined 2D curve, and

(Δx′,Δz′) is the translation to be applied.

Remarks

This is different from the **lstl** option of the **ld** command because **lt** modifies an existing 2D curve. See the example on the following page.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
ld 1
lp2 0 0 .14 .14 .28 .28 .42 .41
.56 .53 .7 .64 .84 .74
.98 .83 1.12 .9 1.26 .95 1.4 .98
1.54 1 1.68 .99 1.82 .97
1.95 .93 2.09 .87 2.23 .79 2.37
.69 2.51 .59 2.65 .47
2.79 .34 2.93 .21 3.07 .07 3.21
-.07 3.35 -.21 3.49 -.34
3.63 -.47 3.77 -.59 3.91 -.69
4.05 -.79 4.19 -.87 4.33
-.93 4.47 -.97 4.61 -.99 4.75 -1
4.89 -.98 5.03 -.95 5.17
-.9 5.31 -.83 5.45 -.74 5.59
-.64 5.72 -.53 5.86 -.41 6
-.28 6.14 -.14 6.28 0;
lscz 1 3
lt 1 1.570796 0
```

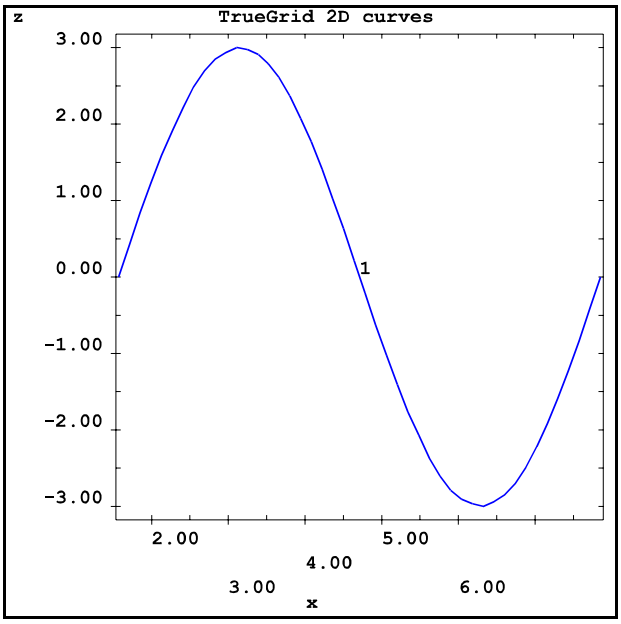


Figure 12 Translate Existing 2D Curve

flcd piece-wise trigonometric load curve

flcd *options ;*

where an *option* can be:

tinit *initial_time*	specify the initial time
tsca *time_scale*	scale the time variable
toff time_offset	translate time variable
finit initial_load	specify the initial load
fsca load_scale	scale the load variable
foff load_offset	translate the resulting curve after scaling
phase angle	specify the phase for a **sine** or **cosine** curve
lp t_1 f_1 t_2 f_2 ... t_n f_n ;	append a polygonal line or curve
sine #_points #_cycles_per_time time	append a sine curve, computed as the sine of the scaled parameter plus the phase
cosine #_points #_cycles_per_time time	append a cosine curve, computed as the cosine of the scaled parameter plus the phase.

Remarks

You specify a number of curve segments, with the **lp**, **sine**, and **cosine** options. The whole load curve is constructed by appending them; each curve begins where the last one ended.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

The initial time, **tinit**, and the initial load, **finit**, are global values. Their last values are the offsets applied to the entire curve. The variables **toff**, **tsca**, **foff**, and **fsca** can be set for each segment. The time variable of a segment is relative to that segment and is assumed to start at zero for each segment.

In the following discussion, the sequences of time and function values are denoted by T_i and F_i respectively. The index *i* is incremented by 1 for each new value in the sequence. For the polygonal line, you specify the time and function values t_i and f_i. Any transformations you specified would change these to the actual time and function values T_i and F_i. The curve is linear between them. For each trigonometric curve, you specify the number of points N, the number of cycles per unit time *cps*, and the time interval Δt covered. For each new *i* value, t_i is incremented by δt=Δt/N. The value of the cosine or sine function gives f_i. In the equations below, **T0** is the untransformed time variable at the end of the last segment and is initially 0.

Thus, after taking into account the other parameters you may specify S **tinit**, **tsca**, **toff**, **finit**, **fsca**, **foff**, and **phase** S the time and function values are:

For a polygonal line (lp):

T_i = tsca * (t_i + T0) + toff + tinit

F_i = fsca * f_i + foff + finit

For a sine curve (sine):

T_i = tsca * (t_i + T0) + toff + tinit

F_i = fsca * sin(*cps* * (t_i + T0) + phase) + foff + finit

For a cosine curve (cosine):

T_i = tsca * (*ti* + T0) + toff + tinit

F_i = fsca * cos(*cps* * (t_i + T0) + phase) + foff + finit

Example

```
flcd 1 phase 180 cosine 30 5 .1
lp .1 1;tinit .1 finit 1
fsca .1 foff 1 phase 0 sine 60
60 .1;
```

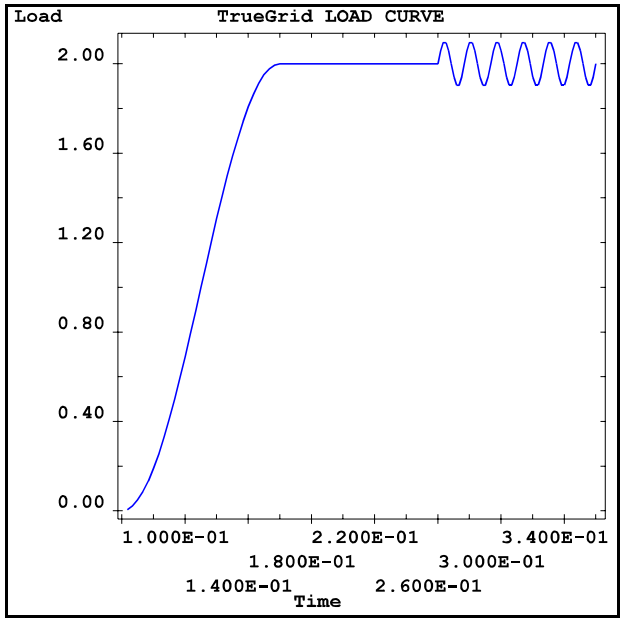


Figure 13 Oscillating Load Curve

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

edgefile identify file used for edge data

edgefile *file_name*

Remarks

The **rln**, **rlns**, and **rseg** commands retrieve data from this file in order to define 2D curves. This feature is archaic but may still be useful.

The file format consists of curve segment data in sets of coordinates as follows. Each set of coordinates begins with a header record with a fixed format:

The first field (I5) is the number of coordinate pairs to follow.

The second field (I5) is the coordinate system flag: 0 for Cartesian coordinates, 1 for the **x′** and **z′**-coordinates to be switched, and 2 for polar coordinates.

The third field (I5) is the reflection flag: 0 for no reflection, 1 to reflect the 2D curve, or 2 to reflect the 2D curve and also keep the unreflected 2D curve.

The fourth field (E10.0) is the Cartesian coordinate scale factor.

The fifth field (E10.0) is the Cartesian coordinate translation in the second coordinate.

The sixth field (E10.0) is the constant second Cartesian coordinate to be used to reflect the 2D curve.

The seventh field (E10.0) is the Cartesian coordinate translation in the first coordinate.

Thereafter are two fields (2E10.0) per record giving each coordinate pair. For Cartesian coordinates, they are interpreted as **x′** and **z′**-coordinates in the **x′z′**-plane. In polar, they are the angle (in degrees) and radius, respectively. Translations, scaling, and reflections are all done in Cartesian coordinates.

The E10.0 fields may be in either floating-point or exponential Fortran format.

This file may contain any number of these 2D curve definitions.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

The following edge file example has 5 curve segments. It is referred to in the examples for the **rln**, **rlns**, and **rseg** commands.

```
   4    0    0 1.0       0.0       0.0       0.0
1.0       1.0
1.5       0.9
2.0       1.0
2.1       1.7
   5    0    0 1.0       0.0       0.0       0.0
2.0       2.0
1.7       2.2
1.0       2.0
0.8       1.7
1.0       1.0
   9    0    0 1.0       2.0       0.0       2.0
1.0       1.0
1.5       0.9
2.0       1.0
2.1       1.7
2.0       2.0
1.7       2.2
1.0       2.0
0.8       1.7
1.0       1.0
   4    0    2 1.0       2.0       3.8       0.0
1.0       1.0
1.5       0.9
2.0       1.0
2.1       1.7
   4    2    1 1.0       0.0       1.0       2.0
135.0     1.5
140.0     1.5
140.0     2.0
145.0     2.1
```

Line number 22 begins the definition of another curve segment with 4 points.

```
   4    0    2 1.0       2.0       3.8       0.0
```

The second number, 0, indicates that the following 4 records will contain *(x, z)* Cartesian coordinates. The next integer, 2, says the curve segment will be reflected across the line at *z=3.8*, specified in the second to last field. It also means that the unreflected curve segment will be included in the segment definition. The next number is 1.0 which means that the coordinates will

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

not be scaled. The fifth field specifies that each **z′**-coordinate will be translated by 2.0. The last field, 0.0, specifies that the **x′**-coordinate will not be translated.

rln import an edge file 2D curve

rln *(no arguments)*

Remarks

Prior to using this command, you must use the **edgefile** command to specify the file which contains the curve data. This command can then be used repeatedly. Each time this command is used, the next 2D curve number is assigned to the next 2D curve segment data found in this file. When all of the data in this file has been used, then the file is automatically closed.

Example

The example edge file in the description on the **edgefile** command above can be converted to 2D curve definitions using the **rln** command. For example, the first curve can be defined with the command:

```
rln.
```

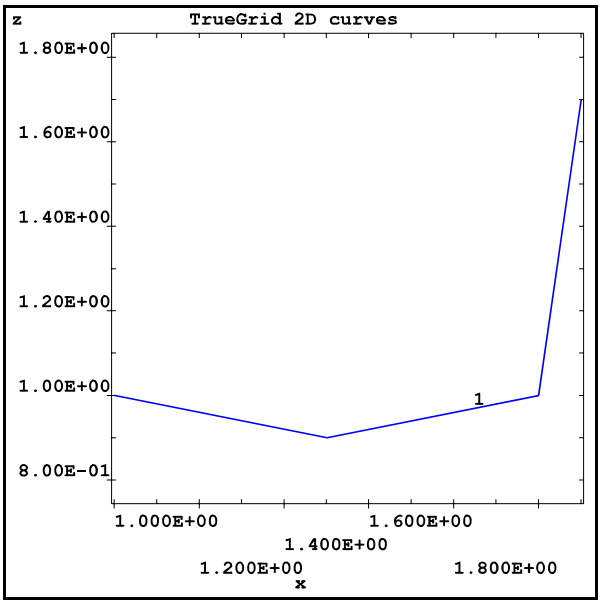


Figure 14 First curve segment

rlns import all edge file 2D curves

rlns *(no arguments)*

Remarks

Prior to using this command, you must use the **edgefile** command to specify the file which contains the curve data. The next 2D curve

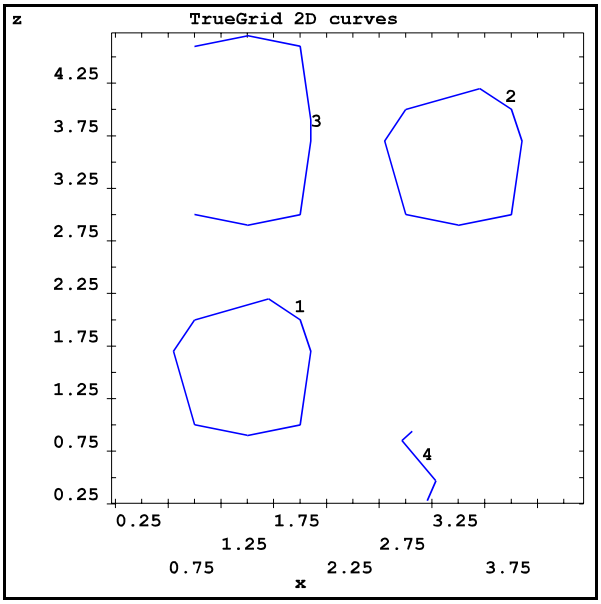


Figure 15 Remaining segments

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

number is assigned to the next 2D curve segment data found in this file. This is repeated until all of the 2D curve segments have be converted to numbered 2D curve definitions. The edge file is then automatically closed.

Example

The example edge file in the description on the **edgefile** command above can be converted to 2D curve definitions using the **rlns** command. For example,

```
rln rseg rlns
```

2. 2D Curve Segment Dictionary

This section describes all the 2D curve segment types. You can use them to construct composite 2D curves with such commands as **ld** and **apld**.

lp2 polygonal line - pairs

lp2 $x'_1 z'_1 x'_2 z'_2 \ldots x'_n z'_n$;

Remarks

lp is the same command, for historical reasons.

Example

```
ld 1 lp2
0 [21/32] .0625 [20.8/32] .125
[20.4/32] .1875 [20/32]
.25 [19.3/32] .3125 [18/32] .375
[16.5/32] .4375 [15/32]
.5 [12.5/32] .5625 [9.6/32] .625
[6/32] .6875 [3.2/32]
.75 [1.8/32] .8125 [1/32] .875
[.2/32] .9375 0;
```

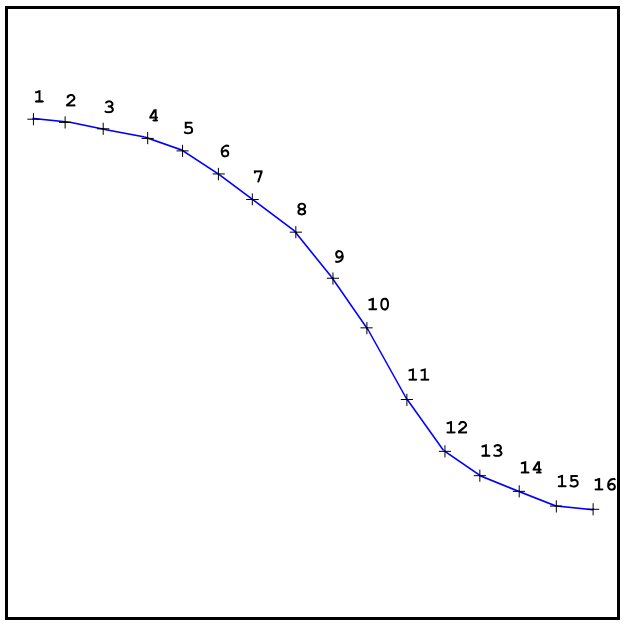


Figure 16 Polygonal line with 16 points

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

lq polygonal line - lists

lq $x'_1\ x'_2 \ldots x'_n\ ;\ z'_1\ z'_2 \ldots z'_m\ ;$

where

$x'_1\ x'_2 \ldots x'_n\ ;\ z'_1\ z'_2 \ldots z'_m\ ;$ are the two coordinate lists.

Remarks

This command takes a list of first coordinates, followed by a list of second coordinates. If one list is shorter than the other, it is automatically extended. See the picture above for the example that follows.

Example

```
ld 1 lq 0.0000 0.0625 0.1250 0.1875 0.2500 0.3125 0.3750 0.4375
0.5000 0.5625 0.6250 0.6875 0.7500 0.8125 0.8750 0.9375;
[21.0/32.0]  [20.8/32.0]  [20.4/32.0]  [20.0/32.0]  [19.3/32.0]
[18.0/32.0]  [16.5/32.0]  [15.0/32.0]  [12.5/32.0]  [9.6/32.0]
[6.0/32.0] [3.2/32.0] [1.8/32.0] [1.0/32.0]
[0.2/32.0] [0.0/32.0];
```

lpil intersect 2 curves

lpil $curve_1\ curve_2$

where $curve_1\ curve_2$ are the 2 2D-curve numbers

Remarks

This command is used to append the intersection point of two previously defined 2D curves. The 2D curve number 3, shown in the example on the next page, uses this feature. The **mazt** is used to specify the tolerance for this intersection.

Example

```
ld 1 lp2
0 [21/32] .0625 [20.8/32] .125
```

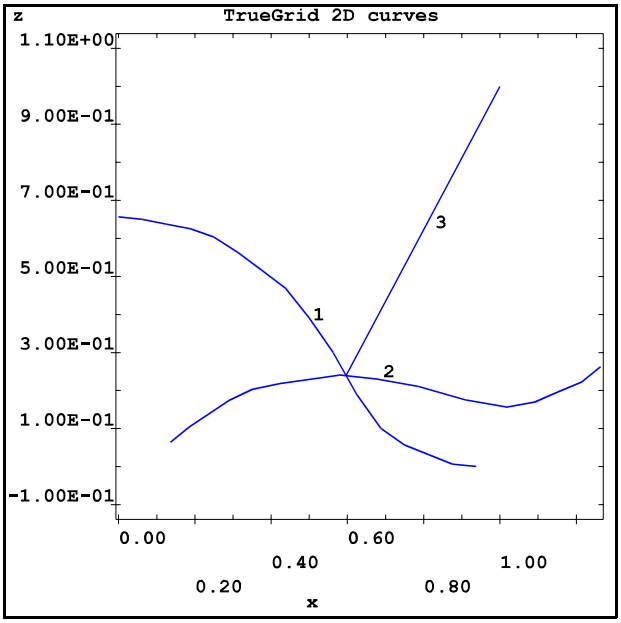


Figure 17 Intersection of 2 curves

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
[20.4/32] .1875 [20/32]
.25 [19.3/32] .3125 [18.0/32.0] .375 [16.5/32] .4375 [15/32]
.5 [12.5/32] .5625 [9.6/32] .625 [6/32] .6875 [3.2/32]
.75 [1.8/32] .8125 [1/32] .875 [.2/32] .9375 [0/32] ;
ld 2 lstl 1 0 0;lrot 2 45 lt 2 .6 -.4
ld 3 lp2 1 1;lpil 1 2;
```

lpta tangent to a circle

lpta $x'_0\ z'_0\ radius$

where

(x'_0, z'_0)	is the center of the circle, and
radius	is the circle's radius, with the sign indicating which side of the circle is to be used.

Remarks

If the *radius* > 0, then the point of tangency is found by moving counterclockwise around the circle from the point formed by the intersection of the circle with the line segment extending from its center to the curves previous point. If the *radius* < 0, then the point is found by moving clockwise. The current 2D curve must have at least one point before this option can be used.

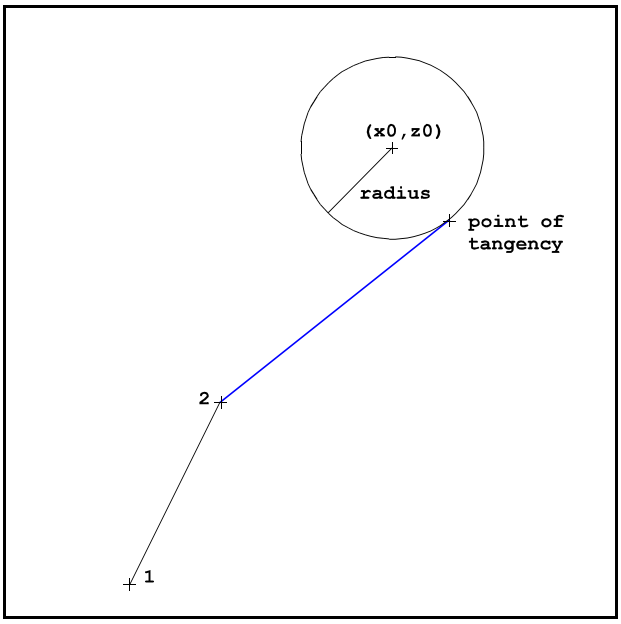


Figure 18 Counterclockwise rotation

Examples

Compare the examples on the following 2 pages. The first has a positive radius, the second has a negative radius.

```
ld 1 lp2 .1 .2 1.1 2.2; lpta 3 5
1;
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
ld 1 lp2 .1 .2 1.1 2.2; lpta 3 5
-1;
```

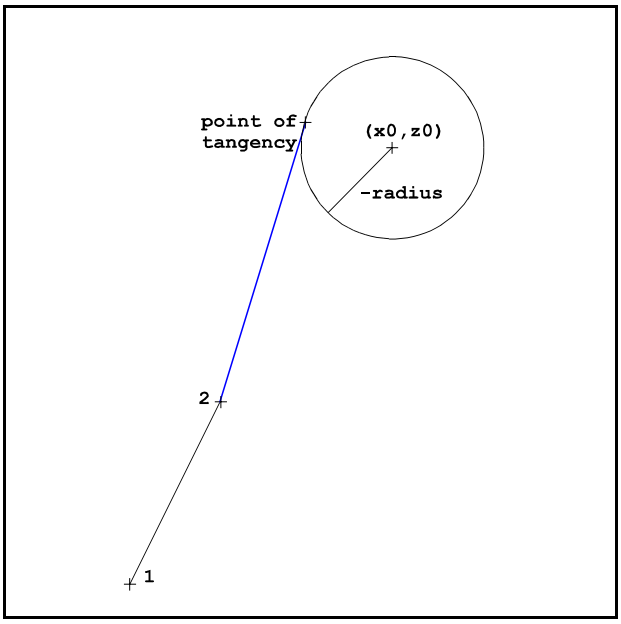


Figure 19 Clockwise rotation

ltas tangent to 2 circles

ltas $x'_0\ z'_0\ flag\ x'_1\ z'_1\ radius$

where

(x'_0, z'_0)	is the center of the first circle,
flag	is 1 for a clockwise arc and -1 for a counter-clockwise arc,
(x'_1, z'_1)	is the center of the second circle, and
radius	is the radius of the second circle.

Remarks

This command appends an arc of a circle joined to a line tangent to two circles. For the first circle, you only specify the center (x'_0, z'_0); the radius is determined by the requirement that it pass through the last point of the current 2D curve. First an arc of the first circle is appended from the last point of the current 2D curve to a line tangent to both circles. This arc travels around the first circle clockwise if *flag* is -1, counterclockwise if 1. Then the tangent line segment from the first circle to the second circle is appended. You specify the center (x'_1, z'_1) and radius of the second circle; the magnitude of *radius* is the radius, and the sign determines which of the possible tangent lines is chosen, where the same convention is used as in the previous command, **lpta**.

The current 2D curve must have at least one point.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

There are generally four lines tangent to two circles. The line is chosen that ensures that the 2D curve's direction will be continuous as we go from an arc of the first circle to the line and then around the second circle (although no arc of the second circle is actually included in the current 2D curve). So *flag*, by determining the direction on the first circle, selects two possible lines. The sign of *radius* determines a direction on the second circle, hence selects one line from those two. For *radius*, a negative number chooses counter-clockwise, and a positive number clockwise.

The four examples that follow show each of the four cases.

Examples

```
ld 1 lp2 1.75 .875 2 1;ltas 2.5 1.3 %r 4 1.9 %s;
```

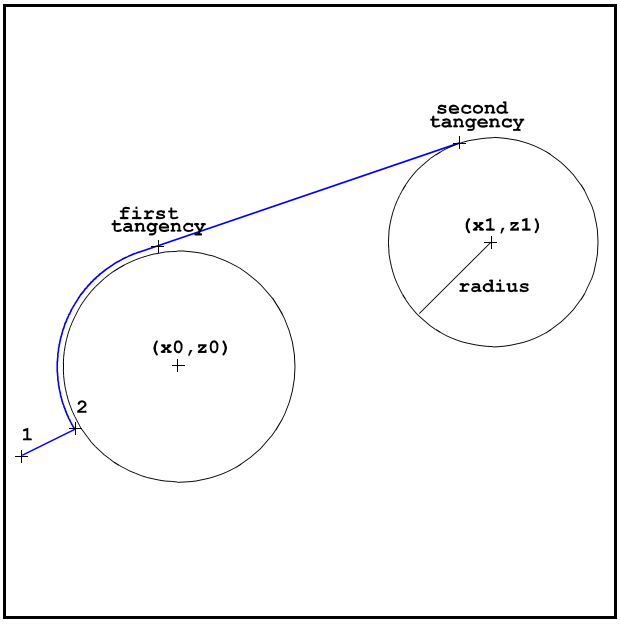


Figure 20 r = -1 , s = .5

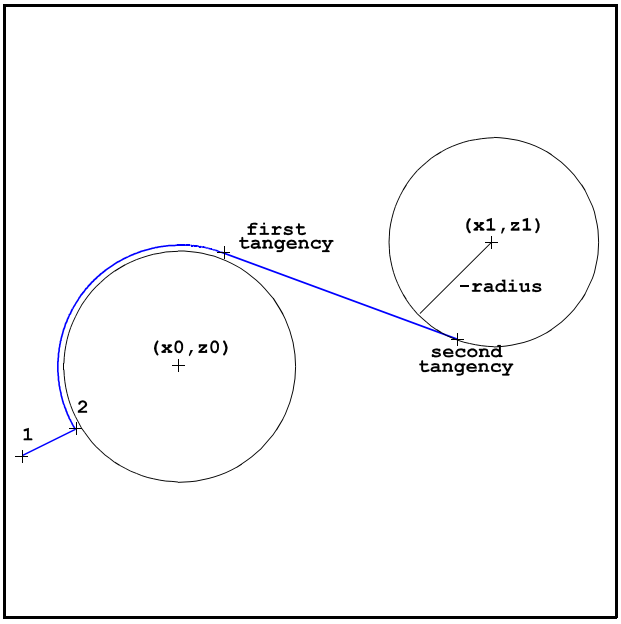


Figure 21 r = -1 , s = -.5

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

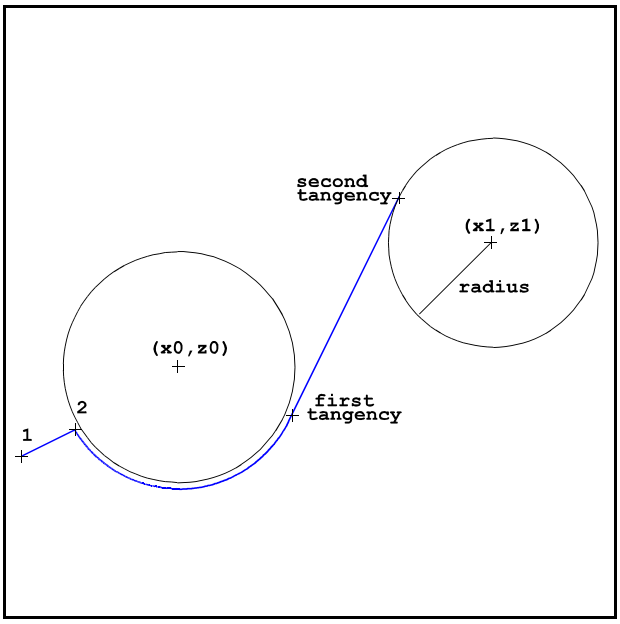


Figure 22 r = 1 , s = .5

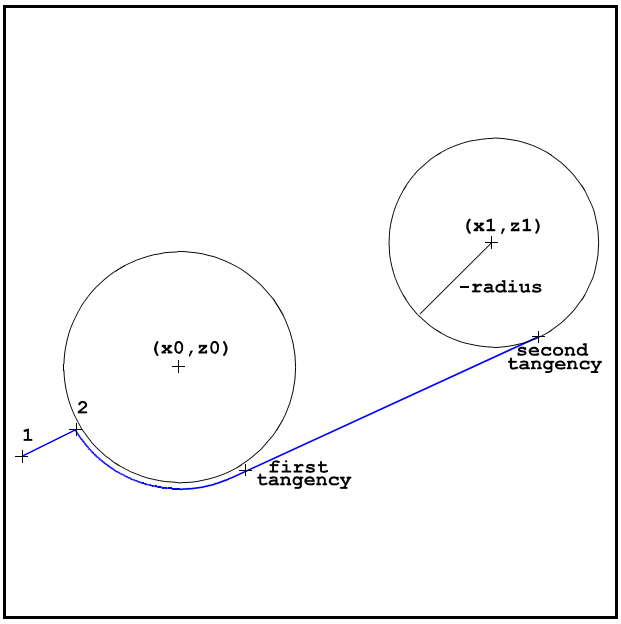


Figure 23 r = 1 , s = -.5

lep elliptic arc

lep $radius_1$ $radius_2$ x'_0 z'_0 θ_{begin} θ_{end} ϕ

where

$radius_1$	is the length of the semi-major axis,
$radius_2$	is the length of the semi-minor axis,
(x'_0, z'_0)	is the center of the ellipse,
θ_{begin}	is the beginning angle of the elliptical arc,
θ_{end}	is the ending angle of the elliptical arc, and
ϕ	is the angle between the major axis and the positive **x**-axis.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

The ellipse is centered at (x'_0, z'_0). The first radius is measured along the axis forming an angle of ϕ degrees with the x-axis of the 2 dimensional local coordinate system, and is referred to as the major axis. The second radius is measured along the axis orthogonal to the first axis, and is referred to as the minor axis. θ_{begin} and θ_{end} are the beginning and ending angles of the arc. θ_{begin} is the angle between the major axis and the line from the beginning point to the center of the ellipse. θ_{end} is the angle between the major axis and the line from the ending point to the center of the ellipse. (The same side of the major axis is used for all three angles). See the example on the following page.

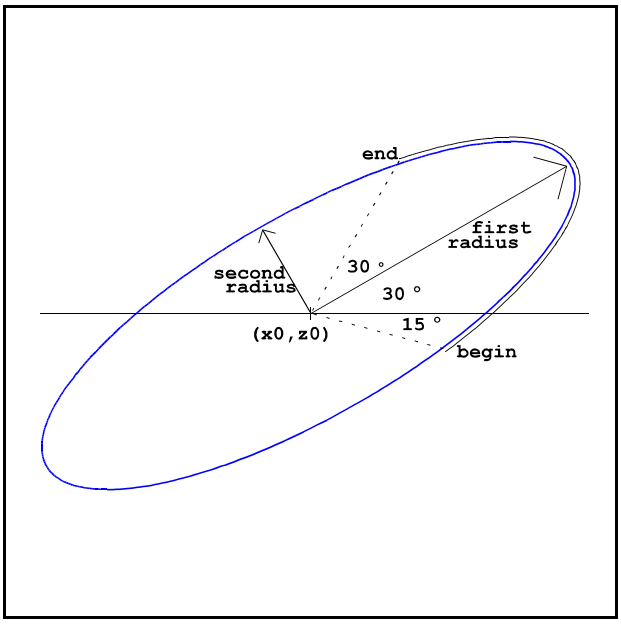


Figure 24 Elliptic Arc

Example

```
ld 2 lep 3 1 1 1 -45 30 30;
```

lod normal offset

lod *curve offset*

where

curve is the ID of a previously-defined 2D curve, and

offset is the distance that this curve is offset in a normal direction.

Remarks

Use this option to take a previously defined 2D *curve* and *offset* it in a normal direction. Positive normals are on the left side of the curve as you move from the beginning to the end. The normals at the endpoints are calculated

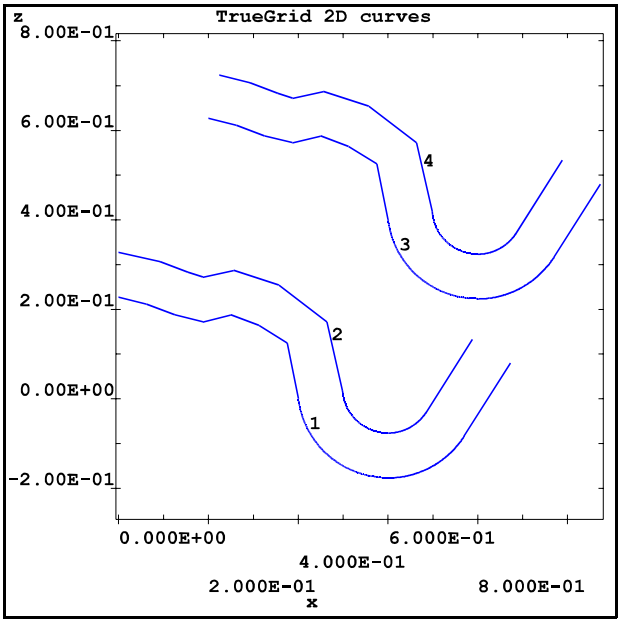


Figure 25 Normal Offset Of 2D Curves

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

independently. If an endpoint of the previously defined 2D curve meets the z′-axis, then the z′-axis is used as the normal direction at that endpoint, causing the offset curve to meet the z′-axis as well. The *offset* is used with the normal directions to determine the endpoints of the new segment. The central difference is used for the normal direction for the interior points. See the next page for an example. In this example, the first curve is on the z′-axis, causing the offset curve 2 to also be on the z′-axis. Since curve 3 is not on the z′-axis, the offset curve 4 is treated differently than curve 2 at the endpoints.

Example

```
ld 1 lp2 0 .2266 .0625 .2109 .125 .1875 .1875 .1719 .25 .1875 .3125
.1641 .375 .125 0.4 0; ltas .6 .025 1 1 0 .15;
ld 2 lod 1 .1 ;
ld 3 lstl 1 .2 .4 ;
ld 4 lod 3 .1 ;
```

lnof normal averaging offset

lnof *curve offset*

where

curve is a the ID number of a previously-defined 2D curve, and

offset is the offset distance.

Remarks

Use this option to take a previously defined 2D curve and offset in a direction using the normal averaging method. For each point in the original curve, both the left and right sided normals are calculated. Then they are averaged. This average normal is used to calculate the direction of the offset. In the example below, compare curves 2 and 3 offset from curve 1 using the **lnof** and **lod** options, respectively.

Example

```
ld 1 lp2 1 0 2 0 2 10 1 10;
ld 2 lnof 1 -1 ;
ld 3 lod 1 -1 ;
```

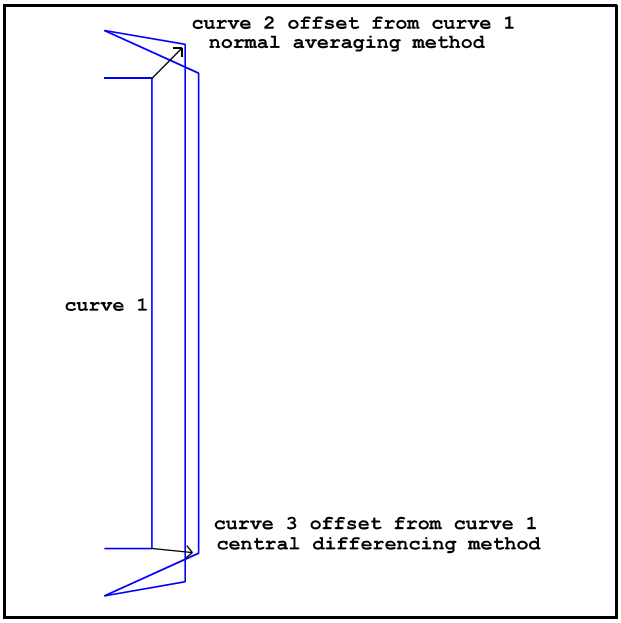


Figure 26 Comparison of two offset methods

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

lfil fillet

lfil $\Theta\, x'_{end}\, z'_{end}\, \Phi\, radius$

where

Θ	is the angle between the positive **x** axis and the line segment which joins the last point of the current curve and the fillet arc,
$(x'_{end}\, z'_{end})$	is the endpoint of this 2D curve segment,
Φ	is the angle between the positive **x** axis and the line segment which joins the fillet arc and the endpoint (x'_{end}, z'_{end}), and
radius	is the radius of the fillet.

Remarks

Use this option to add a fillet made from 2 points and 2 angles to form an arc of a circle of the specified *radius*. This arc is met by a tangent line segment originating at the last defined point in the curve, having the slope specified by the angle Θ. That is, Θ is the angle between the positive x-axis and the line segment. The arc terminates at a point such that if a line is drawn from that point to the point (x'_{end}, z'_{end}), it would be tangent to the circle with the slope specified by the angle Φ. The picture which accompanies the following example should explain everything. The current 2D curve must already have at least one point.

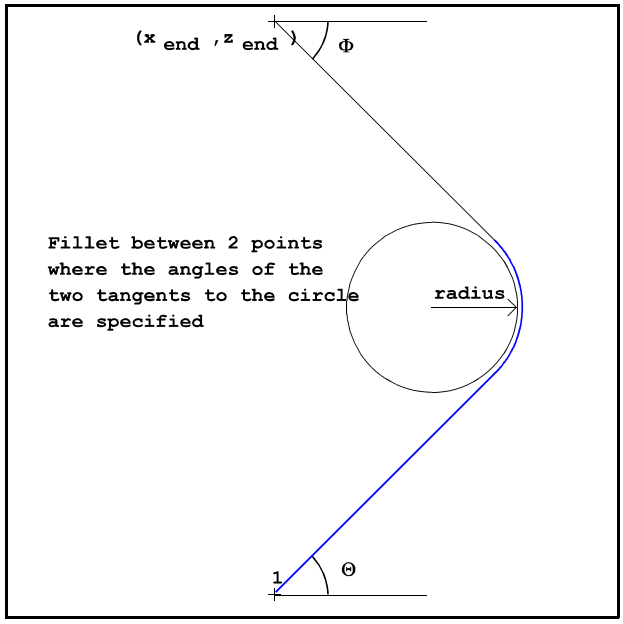


Figure 27 Fillet

Example

```
ld 1 lp2 1 1; lfil 45 1 2 -45 .25 lp2 1 2;
```

lap arc by center and end point

lap $x'_{end}\, z'_{end}\, x'_0\, z'_0$

where

(x'_{end}, z'_{end})	is the end of the line segment which defines the end of the arc, and
(x'_0, z'_0)	is both the center of the circular arc and the beginning of the line segment which defines the end of the arc.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

Use this option to create a circular arc passing through the end of the current curve and terminating at the point of the circle intersecting with the ray containing the point (x_{end}, z_{end}). The circle's center is (x_0, z_0). Thus, the radius of the circle is the distance between (x_0, z_0) and the end of the current curve. The direction (clockwise or counter clockwise) is determined so as to produce the shortest arc between the two endpoints. The current 2D curve must already have at least one point. See the picture on the next page.

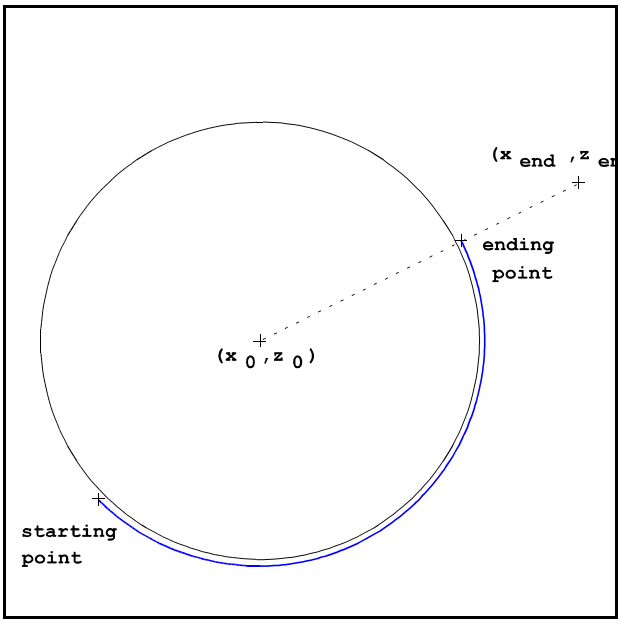


Figure 28 Arc of a circle by point and center

Example

```
ld 1 lp2 1 2;lap 4 4 2 3;
```

lar arc by radius and 2 points

lar x'_{end} z'_{end} *radius*

where

(x'_{end}, z'_{end})	is the end of the arc which is to be created, and
radius	is the radius of this circular arc.

Remarks

Use this option to create a circular arc passing through the end of the current 2D curve and the point (x'_{end}, z'_{end}), with the circle's *radius* also specified. There are two ways to do this. If *radius* is positive, the arc will be counter-clockwise; if negative, the arc will be clockwise. See the example on the following page.

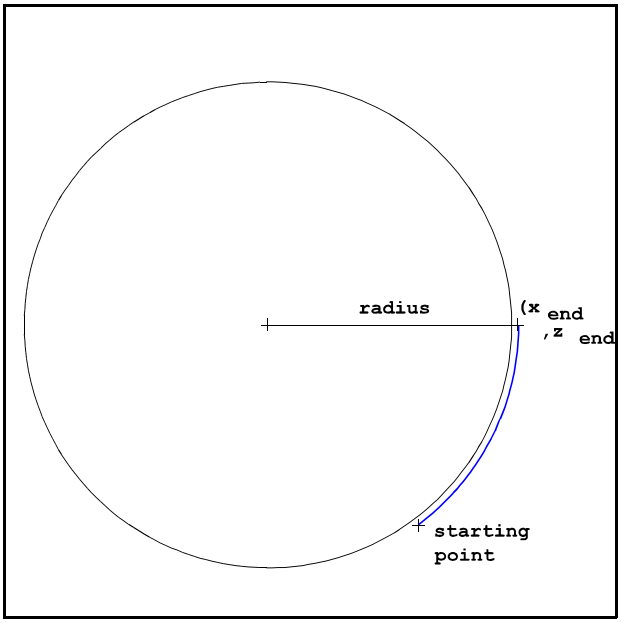


Figure 29 Arc of a circle by radius

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

This option requires that the current 2D curve have at least one point.

Example

```
ld 1 lp2 1 1 ;lar 2 3 2.5;
```

ltp tangent arc by radius and end point

ltp x'_{end} z'_{end} *radius*

where

(x'_{end}, z'_{end})	is the point at which the arc will end, and
radius	is the radius of the arc.

Remarks

Use this option to create a circular arc tangent to the tangent extension of the current curve, ending at a point (**t**angent-to-**p**oint), with the given *radius*. The current 2D curve must have at least two points. The endpoint of the current 2D curve will be extended or cut back to tangentially meet the circular arc. The circular arc will end at the point (x'_{end}, z'_{end}). See the example and picture on the next page.

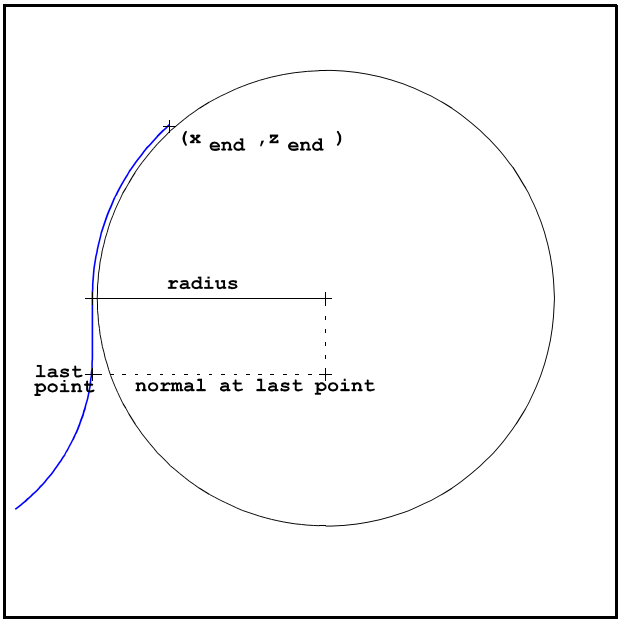


Figure 30 Tangent arc

Example

```
ld 1 lp2 1 1;
lar 2 3 2.5 ltp 3 6 3 ;
```

lpt arc by radius and end tangent line

lpt x'_{begin} z'_{begin} x'_{end} z'_{end} *radius*

where

(x'_{begin}, z'_{begin})	is the end of the specified arc and beginning of the line segment,
(x_{end}, z_{end})	is the end of the line segment, and
radius	is the radius of the circular arc.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

Use this option in order to create a circular arc extended by a tangent line segment (point-to-tangent), where the endpoints of the line segment, (x'_{begin}, z'_{begin}) and (x'_{end}, z'_{end}) respectively, and the radius are specified. The point (x'_{begin}, z'_{begin}) will be modified so that it is the point of tangency. This point is only used to specify the slope of the tangent line segment and it does not have to be at the point of tangency. The arc will begin at the end of the current 2D curve and end at the tangent line segment. The current 2D curve must have at least one point. See the example and the picture on the following page.

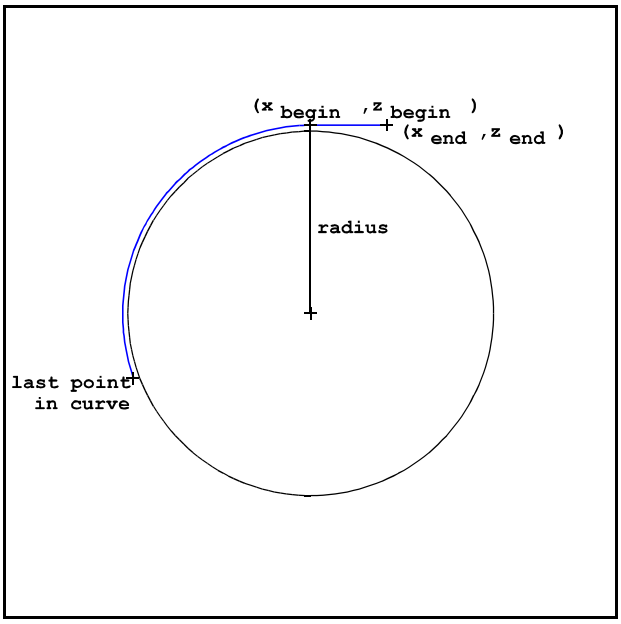


Figure 31 Append arc and tangent line

Example

```
ld 1 lp2 1 1;lpt 2 3 3 3 1.5;
```

lat fillet by radius and end tangent line

lat $x'_{begin}\ z'_{begin}\ x'_{end}\ z'_{end}$ *radius*

where

(x'_{begin}, z'_{begin})	is the point at the beginning of the tangent extension,
(x'_{end}, z'_{end})	is the point at the end of the tangent extension, and
radius	is the radius of the fillet.

Remarks

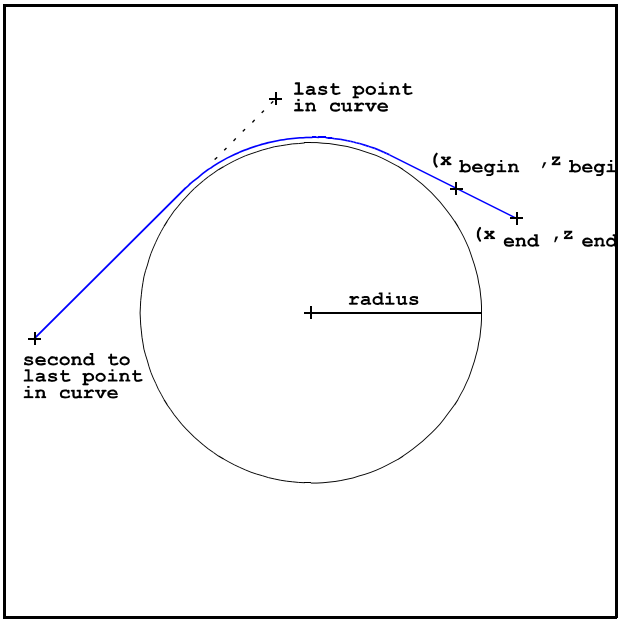


Figure 32 Fillet between two lines

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Use this option in order to append a fillet and a tangent extension by specifying the tangent extension between the points (x'_{begin}, z'_{begin}) and (x'_{end}, z'_{end}) and the *radius* of the fillet (**a**ll **t**angents). It may extend or truncate the last segment of the current 2D curve. The tangent extension may also be extended or truncated. Notice the difference between this and the previous option. This option guarantees a smooth transition. The trade-off is that the current 2D curve must have at least two points.

Example

```
ld 1 lp2 1 1 2 2;
lat 2.75 1.625 3 1.5 .75;
```

lad arc by center and angle

lad $x'_0\ z'_0\ \theta$

where

(x'_0, z'_0)	center of the arc
θ	subtended angle which defines the arc

Remarks

Use this option in order to append an arc of a circle with the specified center (x'_0, z'_0), starting at the last point in the current curve and rotating counterclockwise about a circle, by the specified positive angle θ. A negative angle rotates in a clockwise direction. The current 2D curve must have at least one point. The last point in the current 2D curve is used (with the arc's center) to determine the radius of the arc.

Example

```
ld 1 lp2 2 1 2 2;lad 0 3 45
```

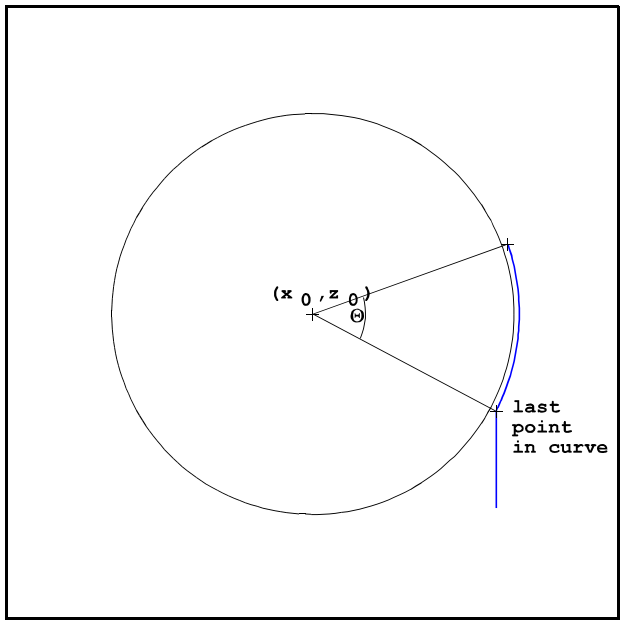


Figure 33 Circular arc

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

lvc point in polar coordinates

lvc *θ radius*

where

θ	is the angle along which the line segment will be drawn, and
radius	is the length of the defined segment.

Remarks

Use this option in order to append a line segment by end point described as an offset from the end of the existing curve. The offset is in polar coordinates. The angle, *θ*, is measured with respect to the positive x-axis and counter clockwise is taken as positive.

Example

```
ld 1 lp2 1 1;
lvc 45 1 lvc -45 1.1
lvc 50 1.2 lvc -55 1.3
lvc 60 1.4 lvc -65 1.5
lvc 70 1.6 lvc -75 1.7
lvc 80 1.8 lvc -85 1.9
lvc 90 2
```

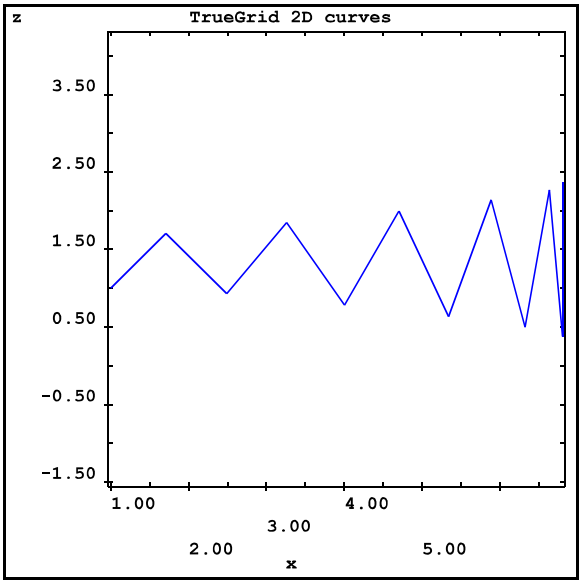


Figure 34 Saw Tooth using polar offsets

lstl translate

lstl *curve Δx′ Δz′*

where

curve	ID number of a previously-defined 2D curve
Δx′	translation distance in the **x′** (horizontal) direction
Δz′	translation distance in the **z′** (vertical) direction

Remarks

Use this command in order to append a translation

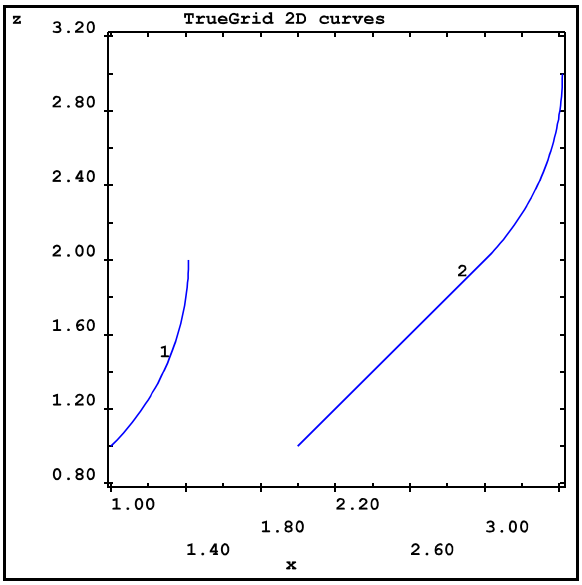


Figure 35 Append a 2D curve

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

of a previously defined and numbered 2D curve to the current 2D curve. See the example and picture on the following page.

Example

```
ld 1 lp2 1 1; lad 0 2 45
ld 2 lp2 2 1; lstl 1 2 1
```

ltbc list of radial coordinates

ltbc θ_{begin} $\Delta\theta$ *scale radii*

where

θ_{begin}	is the first angular coordinate,
$\Delta\theta$	is the angular increment,
scale	is a radial multiplier, and
radii	is the list of radii which will be scaled.

Remarks

Use this option to append points defined in polar coordinates. *Radii* is a list of their radial coordinates. The angular coordinate of the first point is θ_{begin}, and the angular coordinate increases by $\Delta\theta$ for each successive point. Each *radii* is scaled by *scale*. Use the **ltbo** to modify these radii.

Example

The picture which corresponds to the following command file excerpt is shown on the next page.

```
ld 1 ltbc 0 1 1
1.00000 1.02161 1.04201 1.06119
1.07916 1.09592 1.11148 1.12585
1.13904 1.15106 1.16191 1.17163
1.18021 1.18768 1.19406 1.19936
1.20360 1.20682 1.20902 1.21023
1.21049 1.20981 1.20823 1.20577
1.20246 1.19834 1.19344 1.18779
1.18142 1.17436 1.16667 1.15836
1.14948 1.14007 1.13016 1.11980
```

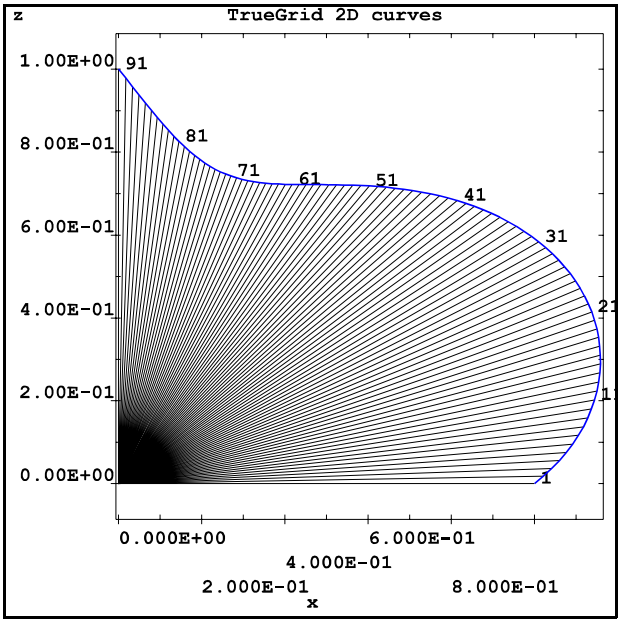


Figure 36 Curve by radial coordinates

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
1.10902 1.09786 1.08637 1.07458 1.06254 1.05028 1.03786 1.02531
1.01268 .999999 .987322 .974687 .962138 .949715 .937462 .925421
.913633 .902140 .890982 .880201 .869837 .859929 .850517 .841639
.833332 .825635 .818584 .812214 .806560 .801656 .797535 .794230
.791770 .790188 .789510 .789766 .790982 .793184 .796396 .800640
.805941 .812316 .819787 .828371 .838085 .848943 .860960 .874148
.888519 .904081 .920843 .938811 .957991 .978387 1.00000;
```

ltbo changes to the list of radial coordinates

ltbo $\Delta radius_1$ $\Delta radius_2$ ... ;

where

$\Delta radius_i$ are the changes for each of the radii.

Remarks

The radial and angular coordinates are initialized (prior to using this command) using the **ltbc** command. The **ltbo** command is subsequently used to change the radial coordinates. These changes are cumulative.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

This example is based on the previous example using the **ltbc** command.

```
ld 2 ltbo
0    3.09017E-02    5.87785E-02
8.09017E-02    9.51056E-02
1.00000E-01    9.51057E-02
8.09018E-02    5.87787E-02
3.09019E-02 0 0 0 0 0 0 0 0 0 0
0   3.09017E-02     5.87785E-02
8.09017E-02    9.51056E-02
1.00000E-01    9.51057E-02
8.09018E-02    5.87787E-02
3.09019E-02 0 0 0 0 0 0 0 0 0 0
0    3.09017E-02    5.87785E-02
8.09017E-02    9.51056E-02
1.00000E-01    9.51057E-02
8.09018E-02    5.87787E-02
3.09019E-02 0 0 0 0 0 0 0 0 0 0
0    3.09017E-02    5.87785E-02
8.09017E-02    9.51056E-02
1.00000E-01    9.51057E-02
8.09018E-02      5.87787E-02
3.09019E-02 0 0 0 0 0 0 0 0 0 0
0    3.09017E-02    5.87785E-02
8.09017E-02 9.51056E-02 .1 9.51057E-02  8.09018E-02 5.87787E-02
3.09019E-02 0;
```

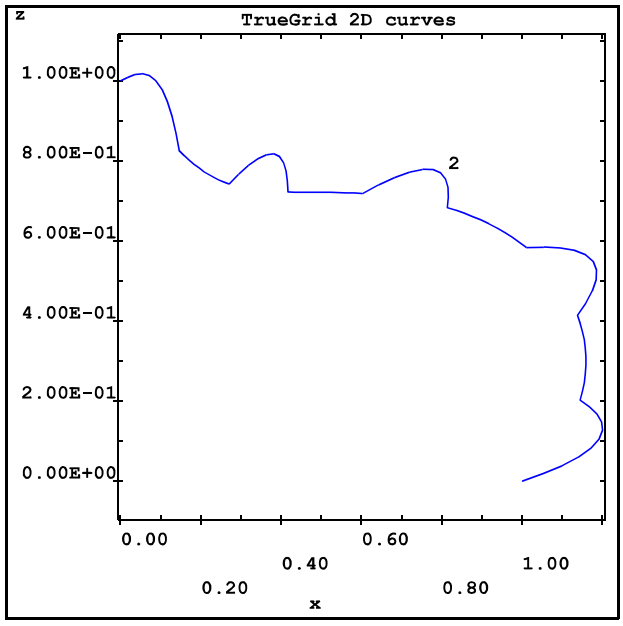


Figure 37 Modified **ltbc** curve

lint interpolation

lint *curve*$_1$ *curve*$_2$ *interpolation*

where

curve$_1$	ID number of the first 2D curve
curve$_2$	ID number of the second curve
interpolation	interpolation parameter

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

Interpolation is an weight factor for the first curve. (1-*interpolation*) is the weight factor for the second curve. In other words, specifying an *interpolation* of 1 will reproduce the first curve and specifying an interpolation of 0 will reproduce the second curve. The interpolation is done by relative arc length. A parameter from 0 to 1, based on the arc length, is used to traverse both curves uniformly. For each value of this parameter, a point is interpolated along the line between the two points associated with the parameter on the two curves. This is done for every point to form the interpolated third curve. See the example on the next page.

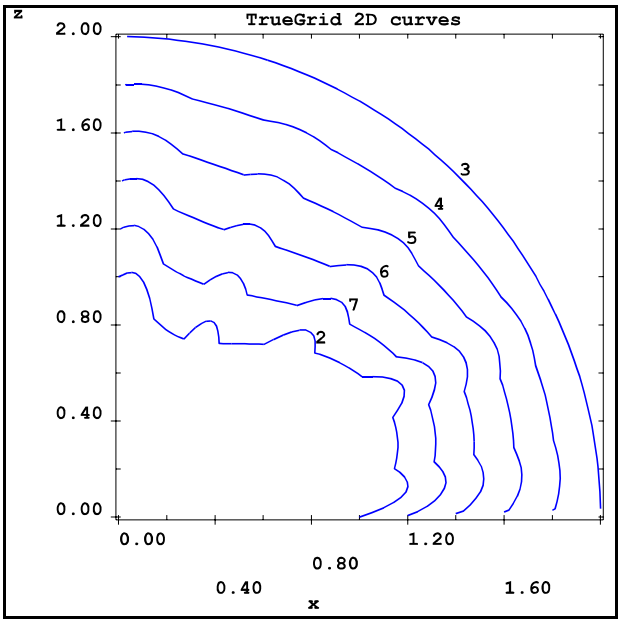


Figure 38 Interpolation Between 2 2D curves

Example

This example is based on the curve defined in the previous example using **ltbo**.

```
ld 3 lep 2 2 0 0 1 89 0;
ld 4 lint 2 3 .2
ld 5 lint 2 3 .4
ld 6 lint 2 3 .6
ld 7 lint 2 3 .8
```

csp2 cubic spline

csp2 *option* $x'_1\ z'_1\ \ldots\ x'_n\ z'_n$

where

option can be

loop	close curve into a smooth loop
First endpoint with zero 2nd derivative	
00	last with zero 2nd derivative
01 *last_dx' last_dz'*	last derivative
02 *last_curve_#*	last match first of other curve
03 *last_curve_#*	last match last of other curve

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

First endpoint derivative

10	*first_dx´ first_dz´*	last with zero 2nd derivative
11	*first_dx´ first_dz´last_dx´last_dz´*	last derivative
12	first_dx´ first_dz´*last_curve_#*	last match first of other curve
13	first_dx´ first_dz´last_curve_#	last match last of other curve

First endpoint match first endpoint of another curve

20	*first_curve_#*	last with zero 2nd derivative
21	*first_curve_# last_dx´last_dz´*	last derivative
22	*first_curve_# last_curve_#*	last match first of other curve
23	*first_curve_# last_curve_#*	last match last of other curve

First endpoint match last endpoint of another curve

30	*first_curve_#*	last with zero 2nd derivative
31	*first_curve_# last_dx´last_dz´*	last derivative
32	*first_curve_# last_curve_#*	last match first of other curve
33	*first_curve_# last_curve_#*	last match last of other curve

$x'_1 z'_1 \ldots x'_n z'_n$ pairs of control points

Example

The polygonal 2D curve number 1 and the spline curve number 2 with natural end derivatives are defined. The spline curve number 3 with loop condition is defined. The loop condition is based on equal coordinates and first derivatives of the start and the end point. Therefore the curve number 3 is smoother than the curve number 2. The start and end point coordinates are (-1,-1). The spline curve number 4 with specified end (-1,-1) and start derivatives (1,1) is defined. The spline curve number 5 is defined by the sane points as curve number 4. It uses start and end derivatives of curve number 4 to form a smooth loop with curve number 4. The spline curve number 6 also uses start and end derivatives of the spline curve number 4, but with different magnitudes. The command file follows:

```
ld 1
 lp2 -1 -1 0 -1.5 1 -1 1.5 0 1 1 0 1.5 -1 1 -1.5 0 -1 -1;
    c polygon from control points

ld 2
 csp2 00
   -1 -1 0 -1.5 1 -1 1.5 0 1 1 0 1.5 -1 1 -1.5 0 -1 -1;;
    c spline with natural derivatives - 00
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
ld 3
 csp2 loop
  -1 -1 0 -1.5 1 -1 1.5 0 1 1 0 1.5 -1 1 -1.5 0 -1 -1;;
     c spline with loop condition - loop

ld 4
 csp2 11 -1 -1 1 1
  -1 -1 0 -1.5 1 -1 1.5 0 1 1 0 1.5 -1 1 -1.5 0 -1 -1;;
     c spline with specified start and end derivatives
     c  -1 -1 and 1 1

ld 5
 csp2 32 4 1 4 1
  -1 -1 0 -1.5 1 -1 1.5 0 1 1 0 1.5 -1 1 -1.5 0 -1 -1;;
     c spline with specified start and end derivatives
     c form curve 4 with magnitude 1      4 1 4 1

ld 5 csp2 32 4 1 4 1
  -1 -1 0 -1.5 1 -1 1.5 0 1 1 0 1.5 -1 1 -1.5 0 -1 -1;;
     c spline with specified start and end derivatives
     c form curve 4 with magnitudes 2 and 4      4 2 4 4
```

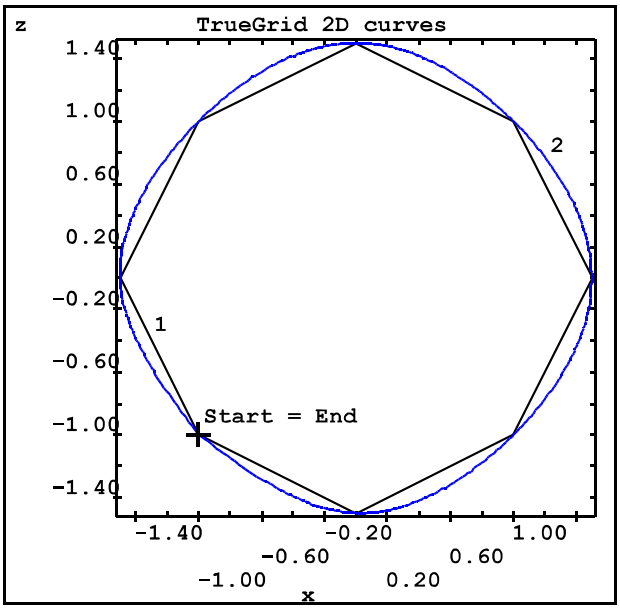


Figure 39 spline with natural end derivatives

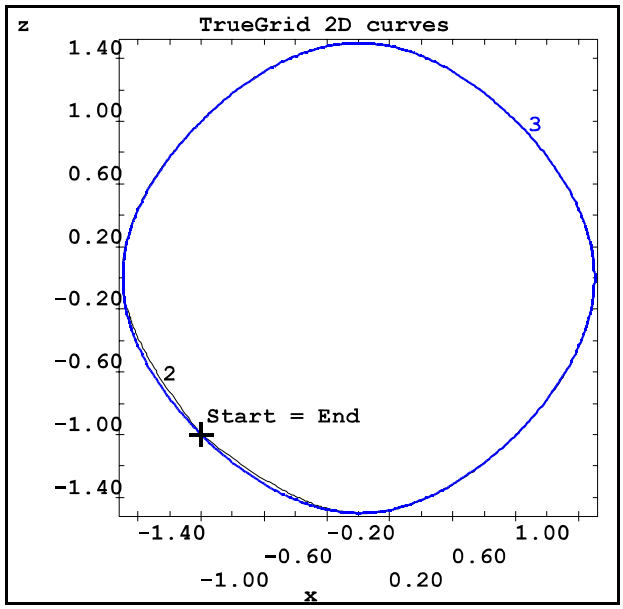


Figure 40 spline with loop condition

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

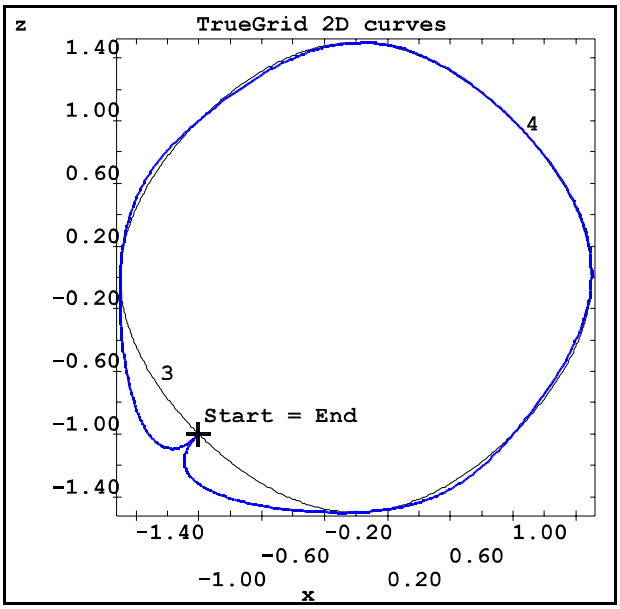


Figure 41 spline with end derivatives

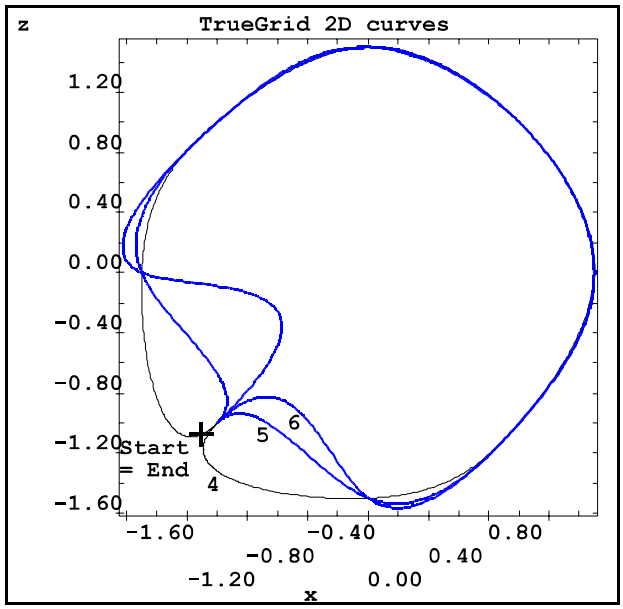


Figure 42 spline with various end derivatives

fws2 Fowler-Wilson cubic spline

fws2 *option convergence mxiniter mxouiter* $x'_1\ z'_1\ \ldots\ x'_n\ z'_n$;

where

option can be

loop close curve into a smooth loop

First endpoint with zero 2nd derivative

00	last with zero 2nd derivative
01 *last_dx' last_dz'*	last derivative
02 *last_curve_#*	last match first of other curve
03 *last_curve_#*	last match last of other curve

First endpoint derivative

10 *first_dx' first_dz'*	last with zero 2nd derivative
11 *first_dx' first_dz' last_dx' last_dz'*	last derivative
12 first_dx' first_dz' *last_curve_#*	last match first of other curve
13 first_dx' first_dz' last_curve_#	last match last of other curve

First endpoint match first endpoint of another curve

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

20	*first_curve_#*	last with zero 2nd derivative
21	*first_curve_# last_dx´ last_dz´*	last derivative
22	*first_curve_# last_curve_#*	last match first of other curve
23	*first_curve_# last_curve_#*	last match last of other curve

First endpoint match last endpoint of another curve

30	*first_curve_#*	last with zero 2nd derivative
31	*first_curve_# last_dx´ last_dz´*	last derivative
32	*first_curve_# last_curve_#*	last match first of other curve
33	*first_curve_# last_curve_#*	last match last of other curve

convergence	convergence tolerance
mxiniter	maximum of inner iterations
mxouiter	maximum of outer iterations
$x'_1\ z'_1 \ldots x'_n\ z'_n$	pairs of control points

Remarks

The Wilson-Fowler algorithm fits cubic spline arcs through the set of given points. The basic idea was to come as close as possible to the true spline (elastica), which minimizes the energy of the curve, with a curve composed of cubic segments which are joined together to achieve continuous tangent and curvature. This was accomplished by introducing a local u,v-coordinate system for each segment, with the independent variable running along the chord joining (x'_i, z'_i) with (x'_{i+1}, z'_{i+1}). The curvature-matching conditions lead to a tri-diagonal nonlinear system of equations, which must be solved iteratively. The difference between derivatives for the *i* and *i+1* point is minimized until the maximal difference is less than *convergence* or until the *maximum number of iterations* is reached.

The recommended value for *convergence* is from 10^{-3} to 10^{-6}. The recommended value for *mxiniter* is 4. The recommended value for *mxouiter* is 8 times the total number of points.

The process does not necessarily converge for every case. When this method does not converge, an informative message is issued with the largest curvature difference. It is useful for simulation of production processes.

The theoretical description of the Wilson-Fowler Spline can be found in:

Cubic Spline, a Curve Fitting Routine. Fowler, Wilson.
Union Carbide Corp., Nuclear Div., Oak Ridge, Tennessee, 1966.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

Two Fowler-Wilson spline curves are defined by sets of points. The points are marked and numbered. The **11** option is used to specify first and second endpoint derivatives, both of the same size `0.001` and `1.` Curve number 2 is randomly perturbed with the normal distribution. The command file and pictures follow:

```
para dx 0.001 mp [sqrt(2)/2]
     d 0.02;
 c parameters dx,mp,d
 c are defined
ld 1 fws2 11 %dx 1 %dx 1
.000001 4 40 1 0 0 1 -1 0 0 -1 1
0;;
 c curve 1
 c - Fowler wilson spline
 c option 11 -
 c specified endpoint deriv.
 c first_dx(%dx) first_dz(1)
 c second_dx(%dx) second_dz(1)
 c convergence  .0000001
 c max.of inner iterations 4
 c max.of outer iterations 40
 c x and z coordinates
 c   of 5 points
```

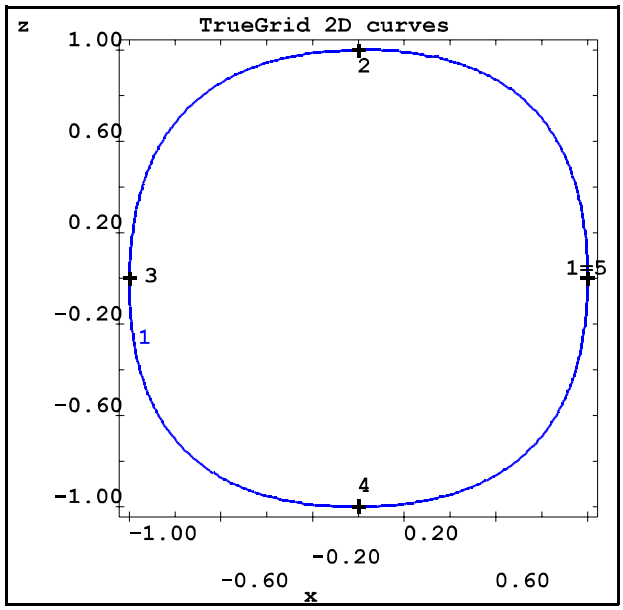


Figure 43 Fowler-Wilson Spline

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
ld 2 fws2 11 %dx 1 %dx 1
  .000001 4 264

 c curve 2
 c - Fowler wilson spline
 c option 11 -
 c specified endpoint deriv.
 c first_dx(%dx) first_dz(1)
 c second_dx(%dx) second_dz(1)
 c convergence  .0000001
 c max.of inner iterations 4
 c max.of outer iterations 264
 c x and z coordinates
 c   of 33 points
```

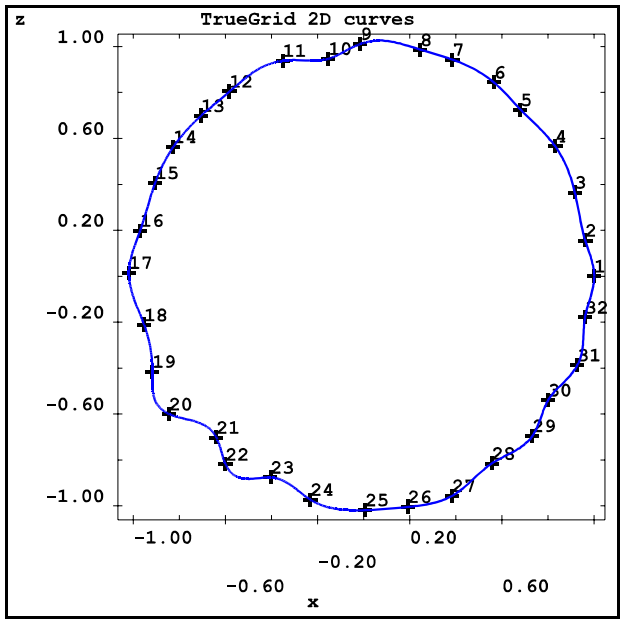


Figure 44 Fowler-Wilson Spline

```
[      1.0          ] [      0.0          ]
[ 0.9772311+%d*norm(0.2138127652)] [ 0.2121777+%d*norm]
[ 0.9238796+%d*norm] [ 0.3826835+%d*norm]
[ 0.8216469+%d*norm] [ 0.5699968+%d*norm]
[ 0.7071068+%d*norm] [ 0.7071068+%d*norm]
[ 0.5699968+%d*norm] [ 0.8216469+%d*norm]
[ 0.3826835+%d*norm] [ 0.9238796+%d*norm]
[ 0.2121777+%d*norm] [ 0.9772311+%d*norm]
[       0.0+%d*norm] [       1.0+%d*norm]
[-0.2121777+%d*norm] [ 0.9772311+%d*norm]
[-0.3826835+%d*norm] [ 0.9238796+%d*norm]
[-0.5699968+%d*norm] [ 0.8216469+%d*norm]
[-0.7071068+%d*norm] [ 0.7071068+%d*norm]
[-0.8216469+%d*norm] [ 0.5699968+%d*norm]
[-0.9238796+%d*norm] [ 0.3826835+%d*norm]
[-0.9772311+%d*norm] [ 0.2121777+%d*norm]
[      -1.0+%d*norm] [       0.0+%d*norm]
[-0.9772311+%d*norm] [-0.2121777+%d*norm]
[-0.9238796+%d*norm] [-0.3826835+%d*norm]
[-0.8216469+%d*norm] [-0.5699968+%d*norm]
[-0.7071068+%d*norm] [-0.7071068+%d*norm]
[-0.5699968+%d*norm] [-0.8216469+%d*norm]
[-0.3826835+%d*norm] [-0.9238796+%d*norm]
[-0.2121777+%d*norm] [-0.9772311+%d*norm]
[       0.0+%d*norm] [      -1.0+%d*norm]
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
[ 0.2121777+%d*norm] [-0.9772311+%d*norm]
[ 0.3826835+%d*norm] [-0.9238796+%d*norm]
[ 0.5699968+%d*norm] [-0.8216469+%d*norm]
[ 0.7071068+%d*norm] [-0.7071068+%d*norm]
[ 0.8216469+%d*norm] [-0.5699968+%d*norm]
[ 0.9238796+%d*norm] [-0.3826835+%d*norm]
[ 0.9772311+%d*norm] [-0.2121777+%d*norm]
[       1.0         ] [       0.0         ];;
```

ctbc polar cubic spline

ctbc *option convergence* θ_{begin} $\Delta\theta$ *scale radii*;

where

option can be

First endpoint with zero 2nd derivative	
00	last with zero 2nd derivative
01 *last_dx´ last_dz´*	last derivative
First endpoint derivative	
10 *first_dx´ first_dz´*	last with zero 2nd derivative
11 *first_dx´ first_dz´ last_dx´ last_dz´*	last derivative

θ_{begin}	first angular coordinate
$\Delta\theta$	angular increment
scale	radial multiplier
radii	list of radii

Remarks

This option is similar to the **csp2** option. The difference is that the spline is calculated in polar coordinates. Points are defined in polar coordinates. *Radii* is a list of their radial coordinates. The angular coordinate of the first point is θ_{begin}, and the angular coordinate increases by $\Delta\theta$ for each successive point. Each *radius* is scaled by *scale*. The derivatives *first_dx´, first_dz´, second_dx´, second_dz´* must be specified in the Cartesian (not polar) coordinate space. Only the direction of the derivative vector is used. The magnitude of the derivative vector is ignored.

Use the **ctbo** option to modify the radii.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

ctbo Modify a Polar Cubic Spline

ctbo *option* $\Delta radius_1$ $\Delta radius_2$ ... ;

where

option can be

First endpoint with zero 2nd derivative	
00	last with zero 2nd derivative
01 *last_dx´ last_dz´*	last derivative
First endpoint derivative	
10 *first_dx´ first_dz´*	last with zero 2nd derivative
11 *first_dx´ first_dz´ last_dx´ last_dz´*	last derivative

$\Delta radius_i$ additive changes to the radii.

Remarks

The spline curve is constructed in the polar coordinate system. All interpolation is done in polar coordinates. The radial and angular coordinates are initialized (prior to using this command) using the **ltbc**, **ctbc**, or **ftbc** option. The **ctbo** option is subsequently used to change the radial coordinates. These changes are cumulative. Only the direction of the derivative vector is used. The magnitude of the derivative vector is ignored.

ftbc Fowler-Wilson polar cubic spline

ftbc *option convergence mxiniter mxouiter* θ_{begin} $\Delta\theta$ *scale radii*;

where

option can be

First endpoint with zero 2nd derivative	
00	last with zero 2nd derivative
01 *last_dx´ last_dz´*	last derivative
First endpoint derivative	
10 *first_dx´ first_dz´*	last with zero 2nd derivative
11 *first_dx´ first_dz´ last_dx´ last_dz´*	last derivative

convergence convergence tolerance

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

mxiniter	maximum of inner iterations
mxouiter	maximum of outer iterations
θ_{begin}	first angular coordinate
$\Delta\theta$	angular increment
scale	radial multiplier
radii	list of radii

Remarks

This option is similar to the **fws2** option. The difference is that the spline is calculated in polar coordinates. Points are defined in polar coordinates. *Radii* is a list of their radial coordinates. The angular coordinate of the first point is θ_{begin}, and the angular coordinate increases by $\Delta\theta$ for each successive point. Each *radius* is scaled by *scale*. The derivatives *first_dx´, first_dz´, second_dx´, second_dz´* must be specified in the Cartesian (not polar) coordinate space. Only the direction of the derivative vector is used. The magnitude of the derivative vector is ignored.

The recommended value for *convergence* is from 10^{-3} to 10^{-6}. The recommended value for *mxiniter* i 4. The recommended value for *mxouiter* is 8 times the number of points.

Example

A Fowler-Wilson polar spline curve is defined by sets of points (see 45). The points are marked and numbered. The **11** option is used to specify first and second endpoint derivatives, both of the same components `0.001` and `1.` The command file and pictures follow:

```
ld 1 ftbc 11 .001 1 .001 1 .001
4 72 10.  10 1 1 .2 .3 .4 .5 .6
.7 .8 .9
```

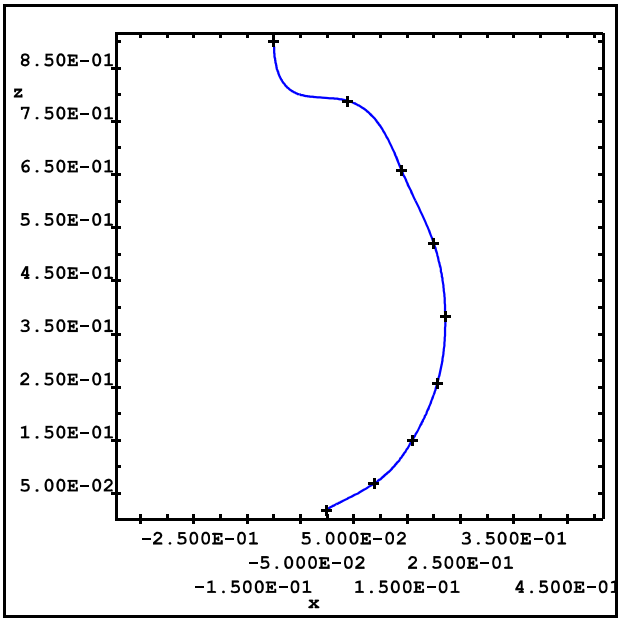


Figure 45 Polar Fowler-Wilson cubic spline

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

ftbo modify Wilson-Fowler polar cubic spline

ftbo *option convergence mxiniter mxouiter* $\Delta radius_1\ \Delta radius_2 \ldots$;

where

option can be

First endpoint with zero 2nd derivative	
00	last with zero 2nd derivative
01 *last_dx´ last_dz´*	last derivative
First endpoint derivative	
10 *first_dx´ first_dz´*	last with zero 2nd derivative
11 *first_dx´ first_dz´ last_dx´ last_dz´*	last derivative
convergence	convergence tolerance
mxiniter	maximum of inner iterations
mxouiter	maximum of outer iterations
$\Delta radius_1\ \Delta radius_2 \ldots$	list of changes in the radii

Remarks

The spline curve is constructed in the polar coordinate system. All interpolation is done in polar coordinates. The radial and angular coordinates are initialized (prior to using this command) using the **ltbc**, **ctbc**, or **ftbc** command. The **ftbo** command is subsequently used to change the radial coordinates. These changes are cumulative. Only the direction of the derivative vector is used. The magnitude of the derivative vector is ignored.

The recommended value for *convergence* is from 10^{-3} to 10^{-6}. The recommended value for *mxiniter* i 4. The recommended value for *mxouiter* is 8 times the number of points.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

The Fowler-Wilson polar spline curve created with the **ftbc** command is modified with the **ftbo** command. The points are marked and numbered. The **10** option is used to specify the first endpoint derivatives, `0.001` and `1.` The command file and pictures follow:

```
ld  1 ftbc 11 .001 1. .001 1
.001 4 72 10 10 1 .1 .2 .3 .4 .5
.6 .7 .8 .9
ld 1 ftbo 10 .001 1. .001 4 72
.1 0 -.1 0 .1 0 -.1 0 .1
```

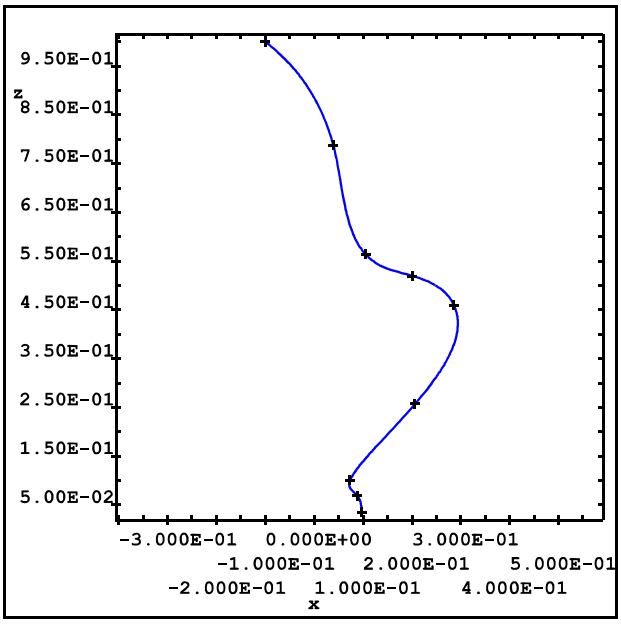


Figure 46 Modified Fowler-Wilson polar

rseg import from edge file

rseg <*no arguments*>

Remarks

Use this option to append the result of reading the next segment of a 2D curve definition in an edge file; see the **edgefile** command. It is required that the **edgefile** command was used first to specified the file containing the curve data. This command can be used repeatedly. Each time this command is used, the next 2D curve segment data found in the edge file will be appended to the present 2D curve. When all of the data in this file has been used, then the file is automatically closed.

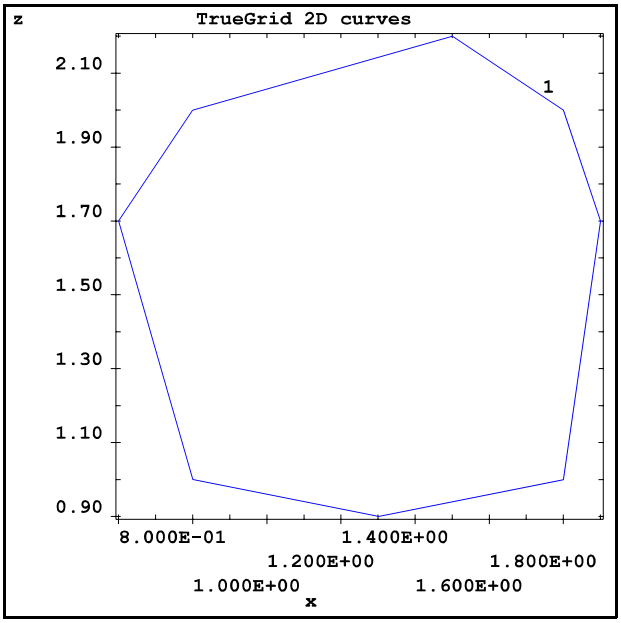


Figure 47 2 segments from edge file

Example

The example edge file in the description on the **edgefile** command above can be converted to 2D curve definitions using the **rseg** command. For example,

```
ld 1 rseg rseg
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

3. 2D Curve Display Commands

These commands control which 2D curves are displayed. Since they are so obvious in what they do, no examples are included in this section.

lcv display a load curve

lcv *load_curve_number*

where

load_curve_number is the number of the load curve to be displayed.

Remarks

If this is the first invocation of a load curve or 2D curve display command, then a new window for the curves is drawn.

lv display all 2D curves

lv *(no arguments)*

Remarks

The command displays all previously defined 2D curves. If this is the first invocation of a curve display command, then a new window is drawn for the curves.

lvi display list of 2D curves

lvi $curve_1\ curve_2 \ldots ;$

where

$curve_i$ is the number of a(the) previously defined 2D curve(s).

Remarks

This command is similar to **dcds**, in that this command allows you to display a list of 2D curves. Other defined 2D curves will not be displayed or will be removed from the display.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

lvs display a sequence of 2D curves

lvs *first_line last_line*

where

first_line	is the first 2D curve to be displayed, and
last_line	is the number of the last curve to be displayed.

Remarks

This command will display every 2D curve whose ID number lies between the 2 specified curve IDs.

4. Create 3D Curves

These commands define, modify, and display information about 3D curves. There are many uses for 3D curves.

You may wish to project a face of the mesh to a surface so that the entire surface is covered. In most cases, you will need to attach the edges of the face to the edges of the surface or to 3D curves that form the shape of the surface boundaries. It is typical to import an IGES model with many surfaces that need to be combined for meshing purposes. This is done with the **sds** option of the **sd** command. Then a face of the mesh can be projected to this composite surface. Once again, to cover the entire surface, attach the edges of the mesh to the boundary of the composite surface which can be formed using the **sdedge** option of the **curd** command or the **coedge** interactive equivalent.

When a surface has more than 180 degrees of curvature, the projection to that surface must be helped by attaching the edge of the mesh to the boundaries or a 3D curve. When there is an interior cusp or feature in the surface that requires a mesh line to follow, a 3D curve can be formed. Then an interior edge of the mesh can be attached to this curve.

If two faces, with a common edge, are projected to two different surfaces, then the common edge will be placed along the intersection of the two surfaces. You can accomplish a similar affect along the edge of the mesh by attaching it to a 3D curve.

When two neighboring faces of the mesh are projected to the same surface, you may wish to control the way the common edge is projected onto the surface by initializing that edge along a 3D curve. This is a common procedure when making fine adjustments to the mesh to improve the mesh quality.

3D curves are also useful when no surface information is available. It was noted above that these curves can be used to define surfaces. It is also possible to place the edges of the mesh along the 3D curves and interpolate the faces of the mesh. In fact, this will be done by default, when no surfaces

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

are specified. However, you may wish to specify the type of interpolation between the edges using the **lin** or **tf** commands.

The **curd** command can be used to begin the definition of a 3D curve. A 3D curve can also be imported from an IGES file by using the **IGES** or **IGESCD** commands. A 3D curve can be used to create 2D curves (**ld3d2d**), 3D curves (**contour**), and surfaces (**sd**). Vertices of the mesh can be placed at points on a 3D curve (**pbs**). Edges of the mesh can be placed along a piece of a curve either initially (**cur**) or permanently (**curf**) or placed along the entire curve (**cure**). **curs** is useful when you want to be able to control the position of the intermediate vertices along a multiple edge.

A 3D curve is built with 3D curve segments. For example, a composite curve from an IGES file is a segmented 3D curve. In many cases, a single 3D curve segment using only one of the options in this chapter is all that is needed for a 3D curve definition. This segmentation feature is something to keep in mind for later when you may wish to make modifications to the geometry by extending a 3D curve definition. Each 3D curve option appends a curve segment to the 3D curve initialized by the last **curd** command. A segment is appended either to the beginning or ending of the existing curve, and the direction of the curve may be switched. This is done to minimize the distance between the endpoint of the existing curve and the endpoint of the appending segment to form one composite curve. The direction of the curve can be checked with the **labels crvpt** command.

You have to specify curves in the correct order, when you are appending or copying curves (**Figure 48** and **Figure 49**)

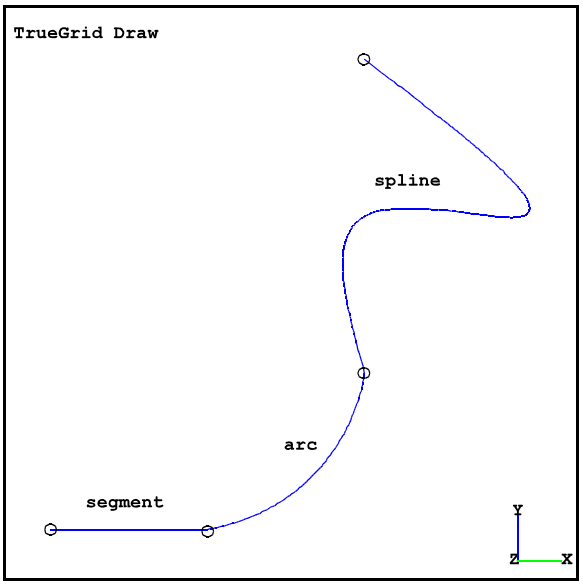


Figure 48 correct order
1 - segment, 2 - arc, 3 - spline

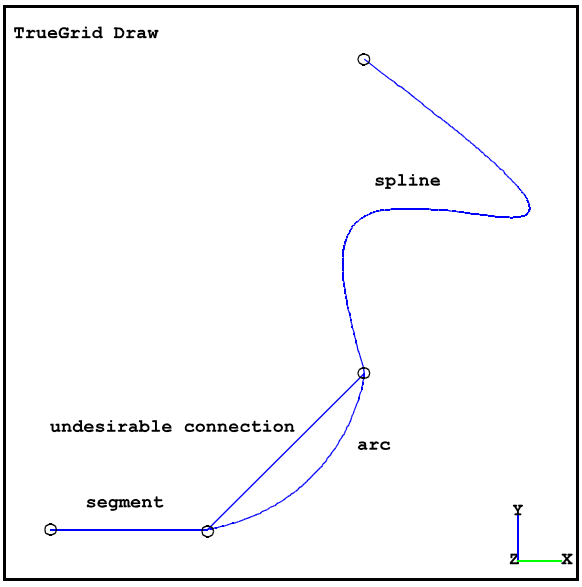


Figure 49 wrong order
1 - segment, 2 - spline , 3 - arc

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

curd 3D curve definition

curd 3d_curve_# type_of_curve curve_data_list
where
type_of_curve and *curve_data_list* can be

igc *iges_curve_#*
sdedge *surface_#.edge_#*
lp3 *x1 y1 z1 ... xn yn zn* ; *trans* ;
contour *surface_point_id_1 surface_point_id_2*
csp3 *option x1 y1 z1 ... xn yn zn* ; *trans* ;
bsp3 *knots x1 y1 z1 ... xn yn zn* ;
nrb3 *knots weights x1 y1 z1 ... xn yn zn* ;
ld2d3d *2d_curve_# system coordinate start end trans* ;
intcur *3d_curve_1 3d_curve_2 interpolate trans* ;
lp3pt *point_id_1 ... point_id_n ; trans* ;
3dfunc *min_u max_u x_expres ; y-expres ; z-expres ; trans ;*
projcur *3d_curve_# surface_# trans ;*
pscur *3d_curve_# surface_# #_iterations tol trans ;*
arc3 *option system point system point system point*
twsurf *surface_1 surface_2 x1 y1 z1 ... xn yn zn ; trans ;*
cpcd *curve_# trans* ;
cpcds *list_curve_# trans* ;
rmseg, and

trans is a product from left to right of the following operators

mx *x_offset*
my *y_offset*
mz *z_offset*
v *x_offset y_offset z_offset*
rx *theta*
ry *theta*
rz *theta*
raxis *angle x0 y0 z0 xn yn zn*
rxy
ryz
rzx
tf *origin x-axis y-axis*
where each of the arguments consist of a coordinate type followed by coordinate

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

information:

rt *x y z* (cartesian coordinates)
cy *rho theta z* (cylindrical coordinates)
sp *rho theta phi* (spherical coordinates)
pt *c.i* (label of a labeled point from a 3D curve)
pt *s.i.j* (label of a labeled point from a surface)

ftf *1st_origin 1st_x-axis 1st_y-axis 2nd_origin 2nd_x-axis 2nd_y-axis*
where each of the arguments consist of a coordinate type followed by coordinate information:

rt *x y z* (cartesian coordinates)
cy *rho theta z* (cylindrical coordinates)
sp *rho theta phi* (spherical coordinates)
pt *c.i* (label of a labeled point from a 3D curve)
pt *s.i.j* (label of a labeled point from a surface)

inv to invert the present transformation
csca scale all coordinates
xsca scale the x-coordinates
ysca scale the y-coordinates
zsca scale the z-coordinates

Remarks

To create a 3D curve, begin by selecting the identification number with the **curd** command. Then select the curve type and associated parameters. Optionally, additional component 3D curve segments can be appended. This is done indefinitely, until a new **curd** command is executed. When a new segment is appended to a 3D curve definition, the data is ordered to best match the endpoint of the last segment appended to the 3D curve definition. The following options are used, within the **curd** command, at any time to append a segment:

Use **igc** to create/append an IGES 3D curve by its sequence number.

Use **sdedge** or **se** to create/append an edge of a defined surface. Each edge is assigned a number. The edge number m of surface number n is identified with the symbol n.m . To view the surface with the numbered edge, use the graphics **label** command with the option **sdedge** or **se**.

Use **arc3** to create/append a 3D arc of a circle passing through 3 points. There are three options. An arc can be created passing from the first to the second to the third point. The second option is the complement of the first option above. That is, the other portion of the circle is used. The third option creates the entire circle.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Use **lp3** to create/append a sequence of points forming a polygonal 3D curve.

Use **lp3pt** to create/append a sequence of surface/curve points forming a polygonal 3D curve.

Use **cpcd** to create/append a copy of a previously defined 3D curve.

Use **cpcds** to create/append a copy of a sequence of previously defined 3D curves.

Use **contour** to create/append a contour segment from an existing surface or curve. The *surface_point_id* is formed from 3 numbers separated by periods. For example 3.10.12 references a point on surface 3, at position i=10, j=12. The *curve_point_id* is formed from 2 numbers separated by a period. For example, 4.1 references the 1st point in 3D curve 4.

Use **csp3** to create/append a 3D cubic spline through a set of ordered points. The end derivatives can be specified or matched.

Use **bsp3** to create/append a b-spline curve formed from an explicit list of knots and control points.

Use **nrb3** to create/append a nurbs curve formed from an explicit list of knots, weights and control points.

Use **intcur** to create/append a curve interpolated between two other 3D curves.

Use **twsurf** to create/append the curve at the intersection of two surfaces. Initial points are required to determine the endpoints and some intermediate points for surfaces with large curvature (greater than 120 degrees). These points need only be approximate.

Use **ld2d3d** to create/append a 2D curve converted to 3D.

Use **3dfunc** to create/append a parameterized function curve with a specified domain.

Use **projcur** to create/append another 3D curve projected onto a surface.

Use **pscur** to create/append another 3D curve projected onto a surface and smoothed.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

igc create/append an IGES curve to a 3D curve

igc *IGES_curve transformations;*

where

IGES_curve is the sequence number of a curve in the current IGES file, and

trans is a transformation

Remarks

A composite IGES curve is treated as a single curve. After the 3D curve is defined, the **saveiges** and **useiges** commands should be used for fast retrieval of the IGES curve in subsequent sessions. See their command descriptions for more detail. A transformation is formed by a sequence of simple transformations such as rotate and translate. For a complete list of these simple transformations, see the **lct** command.

sdedge create/append curve by surface edge

sdedge *surface_#.edge_#*

where

surface_# is the surface ID number of a previously defined surface, and

edge_# is the number of the desired edge. The edge numbers range from 1 to 4.

Remarks

Use **sdedge** or **se** to create/append a 3D curve from an edge of a defined surface. Each edge is assigned a number. To view the surface with the numbered edge, use the graphics **labels** command with the option **sdedge** or **se**. It is a good idea to zoom in tight on the desired edge before turning on these labels, however, particularly if you are using read-in IGES surfaces. You cannot select the edge of an infinite surface such as a cone.

There is an advantage to converting an edge of a surface to a 3D curve and attaching an edge of the mesh to that 3D curve, instead of attaching the same edge of the mesh directly to the edge of the surface using the **edge** command. When using the **edge** command, all interior vertices of the edge of the mesh will be interpolated as a function of the two end vertices. This implies that you have no freedom to position these interior vertices. In contrast, when attaching a multiple edge to a 3D curve, you have freedom to position each vertex, independently, along the 3D curve. If you are attaching a simple edge, one with no interior vertices, then there is no obvious preference, unless, in the future reuse of this model you should choose to insert a partition (**insprt**). If the edge was attached to an edge of the surface using the **edge** command, you will not be able to position the new interior vertex.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

The interactive feature called **coedg** is an easy way to sequence many surface edges together to form a composite curve. The tsave file will record the **coedg** command as a sequence of **se** options in a **curd** command.

The order of the edges in a sequence of **sdeges** options determines the way the edges are sewn together. If there is a gap between two consecutive edges, the gap will be filled with a line segment.

Example

```
sd 1 function 0 10 0 10 u*u ; u*v ; v*v ; ;    c  surface 1
sd 2 function 2 10 10 15 u*u ; u*v ; v*v ; ;    c  surface 2
curd 1 sdedge 1.2 sdedge 2.2   c curve 1 definition from 2 edges
```

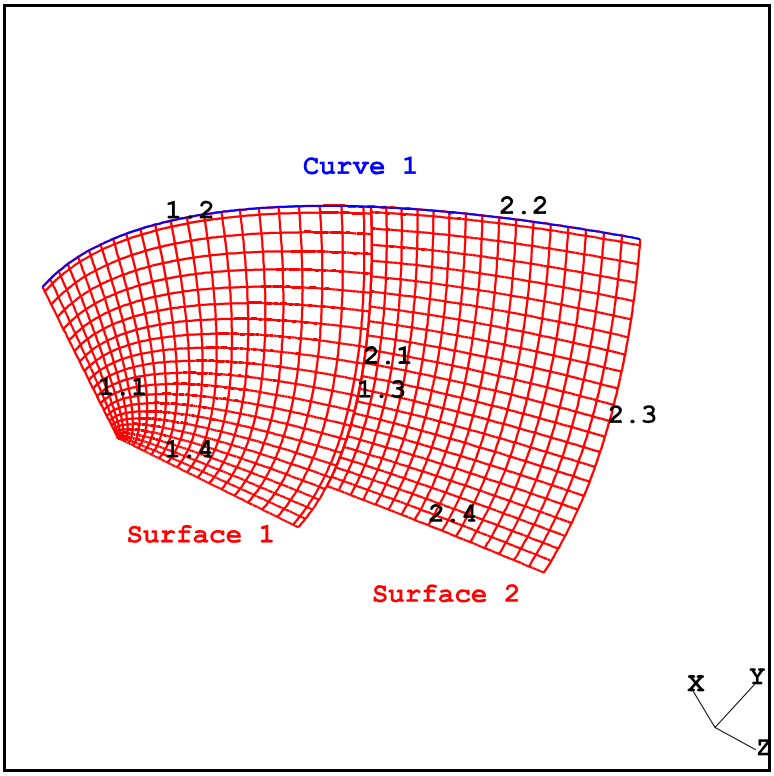


Figure 50 sdedge example

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

lp3 create/append polygon of segments

lp3 *x1 y1 z1 ... xn yn zn* ; *trans* ;

where

x1 y1 z1 ... xn yn zn	are triplets of coordinates of points forming a polygon, and
trans	is a transformation.

Remarks

Use the **lp3** option to create/append a 3D curve by defining a series of points which are used to define line segments. A transformation is formed by a sequence of simple transformations such as rotate and translate. For a complete list of these simple transformations, see the **lct** command.

Example

```
curd 1 lp3 0 0 0 1 1 0 3 1 0 5 0 0 6 0 0;;
```

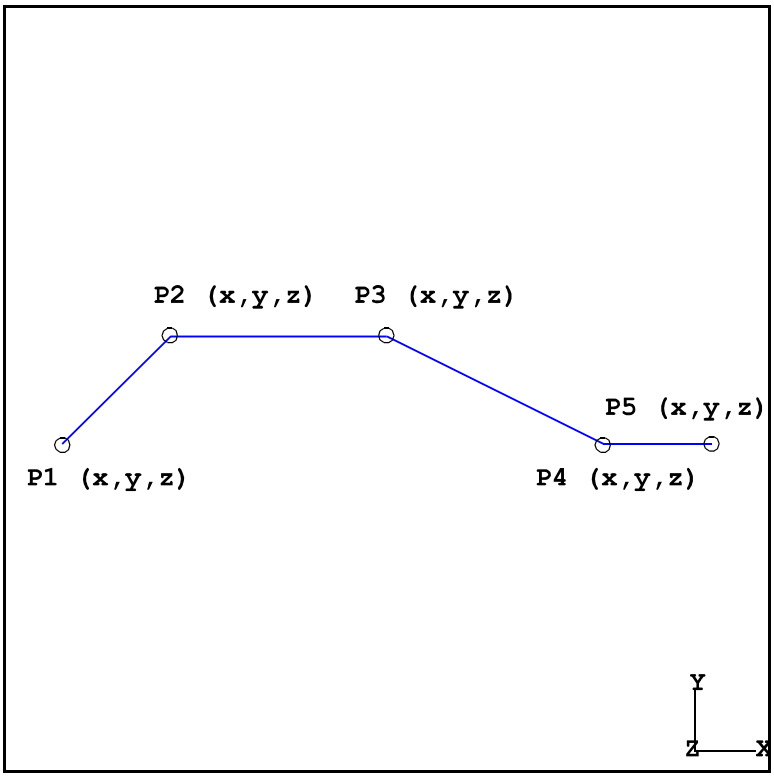


Figure 51 lp3 example

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

contour create/append a contour line to a 3D curve

contour *point*$_1$ *point*$_2$

where

point$_1$ and *point*$_2$ identify points of a surface or 3D curve.

Remarks

The term contour is used to refer to a curve connecting two points of a surface or 3D curve. These are points used in the tessellation of the surface or curve and assigned point identifiers. To find the point identifiers, label the points of the surface or curve: either issue **labels sdpt** or **labels crvpt** or use the **Labels** button in the Environment Window. Then you will see point identifiers of the form *s.i.j* on a surface, or *c.i* on a curve. These points along a 3D curve are numbered sequentially. A similar scheme is used in two directions for a set of points on a surface. The surface is drawn as a grid of contour lines connecting some of these points on the surfaces (see **sdint**). When choosing a contour from a surface, the two points, *point*$_1$ and *point*$_2$, should differ only in their *i* values or only in their *j* values to extract an existing surface contour. *I* or *j* equal to **zero** means the maximum value of *i* or *j*, respectively. If both *i* and *j* are different in the two points, a smooth curve along the surface will be selected to connect the two points.

All of the coordinates of the contour line are written to the **tsave** file.

Example

The contour curve is created on the ruled surface.

```
ld 1 lp2 1 1 -1 1 -1 -1 1 -1 1 1;
ld 2 lp2 1 1;
lad .5 .5 360 ;
sd 1 rule2d 1 1 2 2 ;    c ruled surface definition
curd 1
contour 1.4.1 1.4.0 ;    c curve defined by first point 1.4.1 and
                       c last point 1.4.0 on contour 4 of surface 1
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

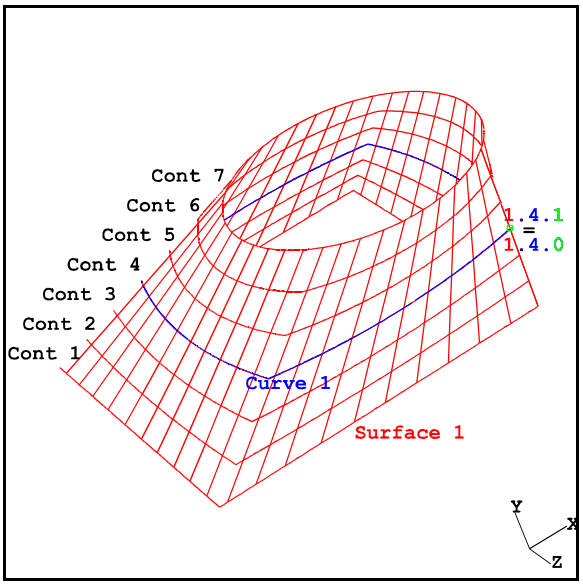


Figure 52 Contour Curve on Ruled Surface

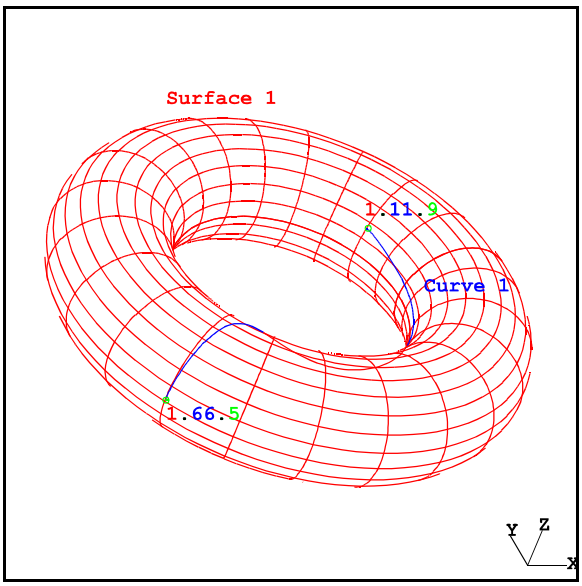


Figure 53 Contour Curve on Torus Surface

Example

You can also connect points from 2 different contours (66 and 11 in the i-index, 5 and 9 in the j-index). A curve is formed by seeking a short path (not necessarily the shortest path) between the first and last point. In a periodic surface, such as a torus, it will choose the shortest periodic path, but the smoothing algorithm may not converge to the to exact shortest path.

```
sd 1 ts 0 0 0 0 0 1 3 0 1           c torus surface definition
curd 1
contour 1.66.5 1.11.9;   c curve 1 defined by first
                         c point 1.66.5 and last point 1.11.9
```

csp3 create/append a 3D cubic spline curve

csp3 *option x1 y1 z1 ... xn yn zn* ; *trans* ;

where

option can be

loop close curve into a smooth loop

First Endpoint Natural Derivative

00 natural derivative for the Last Endpoint

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

01 *last_dx last_dy last_dz* - derivatives at the Last Point
02 *#_curve_2 derivative_mag_2* - match First Endpoint of Another Curve
03 *#_curve_2 derivative_mag_2* - match Last Endpoint of Another Curve

First Endpoint Specify Derivative
10 *first_dx first_dy first_dz*
11 *first_dx first_dy first_dz last_dx last_dy last_dz*
12 *first_dx first_dy first_dz #_curve_2 derivative_mag_2*
13 *first_dx first_dy first_dz #_curve_2 derivative_mag_2*

First Endpoint Match First Endpoint of Another Curve
20 *#_curve_1 derivative_mag_1*
21 *#_curve_1 derivative_mag_1 last_dx last_dy last_dz*
22 *#_curve_1 derivative_mag_1 #_curve_2 derivative_mag_2*
23 *#_curve_1 derivative_mag_1 #_curve_2 derivative_mag_2*

First Endpoint Match Last Endpoint of Another Curve
30 *#_curve_1 derivative_mag_1*
31 *#_curve_1 derivative_mag_1 last_dx last_dy last_dz*
32 *#_curve_1 derivative_mag_1 #_curve_2 derivative_mag_2*
33 *#_curve_1 derivative_mag_1 #_curve_2 derivative_mag_2*

x1 y1 z1 ... xn yn zn - triplets of coordinates of Control Points

Theory

A cubic spline curve is composed from segments (**Figure 54**). Segment *i* is described by parametric algebraic equations:
where *t* is an independent variable, and *a,b,c,d,e,f,g,h,j,k,l,m* are unknown constants. There are 12

$$x(t) = a_i t^3 + b_i t^2 + c_i t + d_i$$
$$y(t) = e_i t^3 + f_i t^2 + g_i t + h_i$$
$$z(t) = j_i t^3 + k_i t^2 + l_i t + m_i$$

unknown constants for segment *i*. Suppose, there are *n* segments in the cubic spline. This produces *12n* unknowns. There are *6n* constraints since the endpoints of each section pass through the control points. There are *6n-6* additional constraints imposed so that the curve has equal left and right first

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

derivatives at the interior control points.

You can choose the derivatives at the end points to impose the remaining 6 constraints. You have 2 possible choices:

- **first derivatives** at the Endpoints
- **natural derivatives** at the Endpoints (second derivatives = 0)

A transformation is formed by a sequence of simple transformations such as rotate and translate. For a complete list of these simple transformations, see the **lct** command.

Example

You can insert control point in between all ready defined control points from **Figure 54**. The result is shown in **Figure 55**.

```
curd 1
csp3 00 c natural derivative
0 1 0
1 0 0
2 1.3 0
3 0 0
3 0.8 0  c inserted point
4 1 0;;;
```

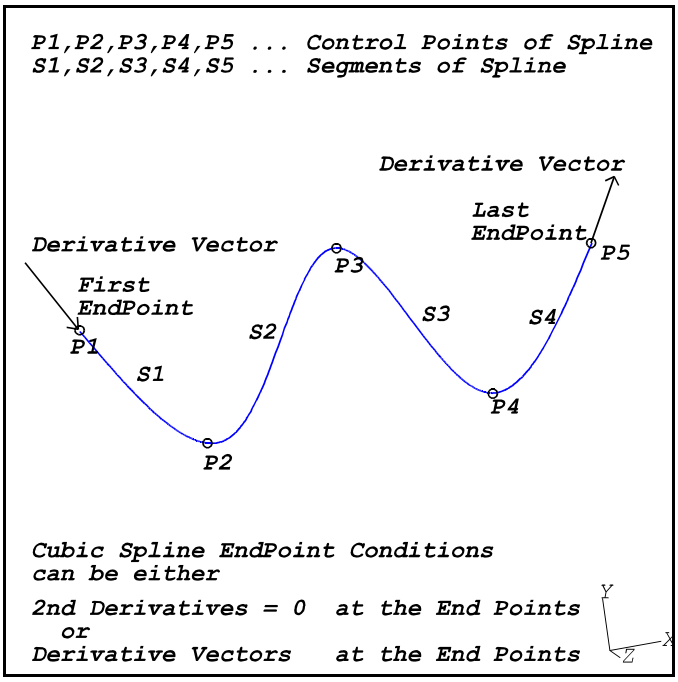


Figure 54 Spline Curve

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

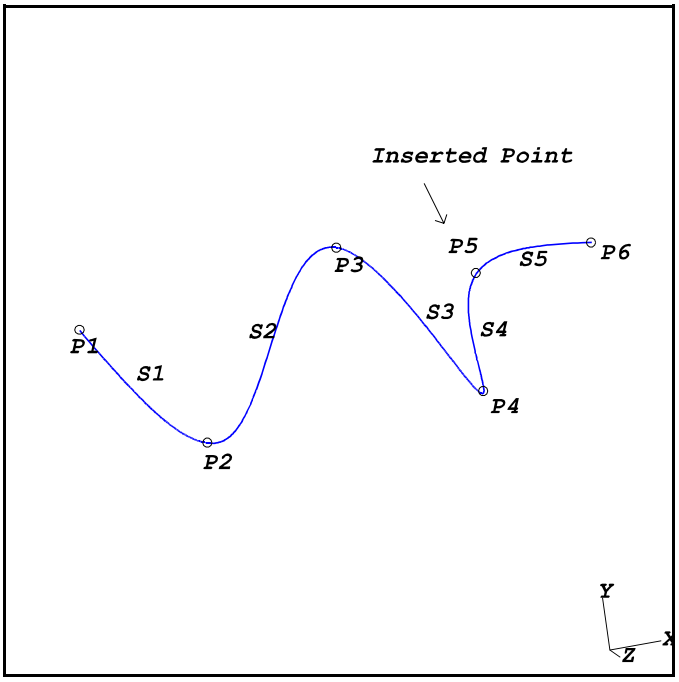


Figure 55 Inserting a Cubic Spline Control Point

Example

You can define a derivative condition for the previously defined control points in **Figure 54**. The result is shown in **Figure 56**.

```
curd 1
csp3 10 1 3 0 c derivatives
0 1 0
1 0. 0
2 1.3 0
3 0 0
4 1 0 ;;;
```

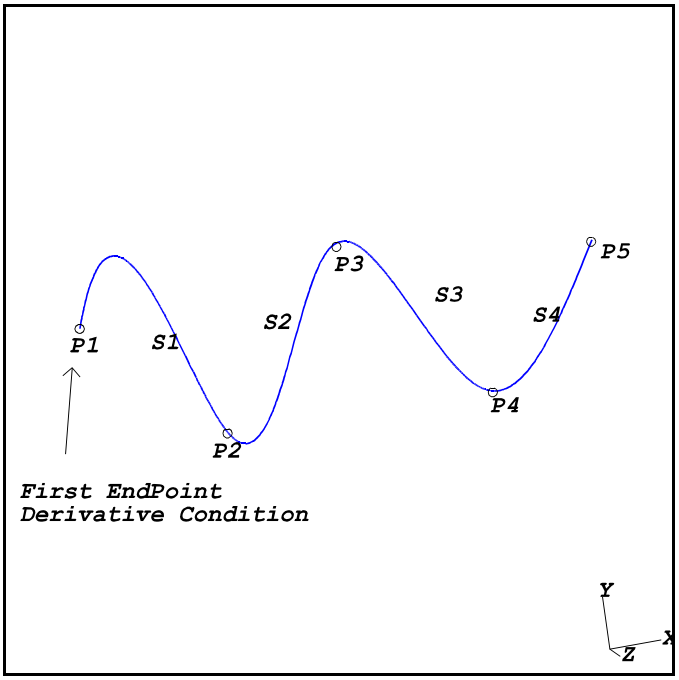


Figure 56 Endpoint Derivative Condition

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

bsp3 create/append a B-Spline curve

bsp3 $knot_1 \ldots knot_n$; $x_1\ y_1\ z_1 \ldots x_k\ y_{k\ zk}$; *trans* ;

where

$knot_1 \ldots knot_n$ are the parametric coordinates of *n* knots

Remarks

The multiple knots have to be specified for the starting and ending point of the curve. The number of multiple knots for the starting and ending point determines the order of the curve (e.g. 2 - linear, 3 - quadratic, 4 - cubic, etc.). The multiplicity has to be the same for the starting and ending point.

$x_1\ y_1\ z_1 \ldots x_k\ y_k\ z_k$ are triplets of coordinates of control points.

where $k = n$ - *multiplicity*

A transformation is formed by a sequence of simple transformations such as rotate and translate. For a complete list of these simple transformations, see the **lct** command.

Example

```
curd 2 bsp3
     0 0 0 1 2 3 3 3; c knots
     0 0 0           c control
     1 1 1           c points
     2 1 1
     2 0 1
     3 2 1; ;

c bspline curve definition
c 4 knots
c 0 0 0 multiple knot
c 3 3 3 multiple knot
c 5 control points
c P1...P5
```

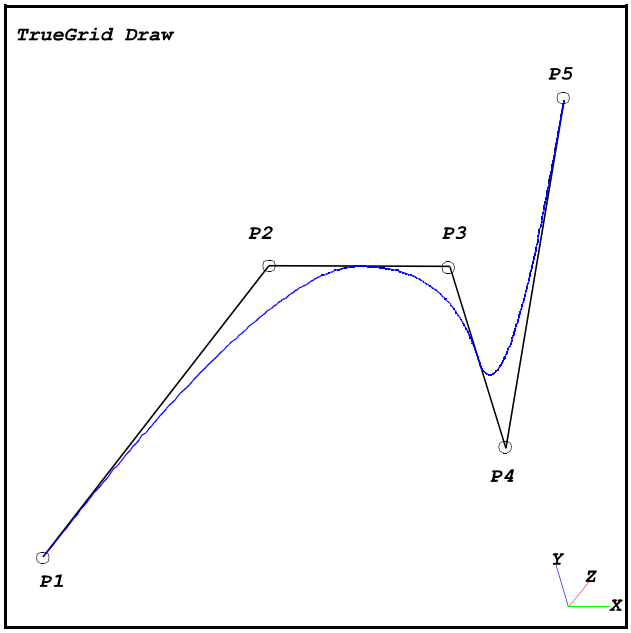


Figure 57 B-spline curve

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

nrb3 create/append a NURBS curve

nrb3 $knot_1$ ... $knot_n$; $weight_1$ ... $weight_k$; $x_1\ y_1\ z_1$ ... $x_k\ y_k\ z_k$; *trans* ;

where

$knot_1$ ... $knot_n$	are parametric coordinates of n knots
$weight_1$ ... $weight_k$	are weights for each control point
$x_1\ y_1\ z_1$ ... $x_k\ y_k\ z_k$	are triplets of coordinates of control points.

where $k = n$ - *multiplicity*.

Remarks

The **multiple knots** have to be specified for the starting and ending point of the curve. The number of multiple knots for the starting and ending point determines the order of the curve (e.g. 2 - linear, 3 - quadratic, 4 - cubic, etc.). The multiplicity has to be the same for the starting and ending point.

This command creates/appends a NURBS curve formed from an explicit list of knots, weights and control points.

You can shape the curve by changing the weights (**Figure 58**).

You can specify discontinuity at the control point by multiple internal knots (**Figure 59**).

A transformation is formed by a sequence of simple transformations such as rotate and translate. For a complete list of these simple transformations, see the **lct** command.

Example

```
curd 2 nrb3
 0 0 0 1 2 4 5 5 5;  c knots
.1 1 1 5 1 1;        c weights
     0 1 0           c control
     1 0 0           c points
     2 1 0
     3 4 0
     4 2 0
     5 3 0;;
c curve definition - 2
c nurbs
c 0 0 0 multiple knot
c 5 5 5 multiple knot
```

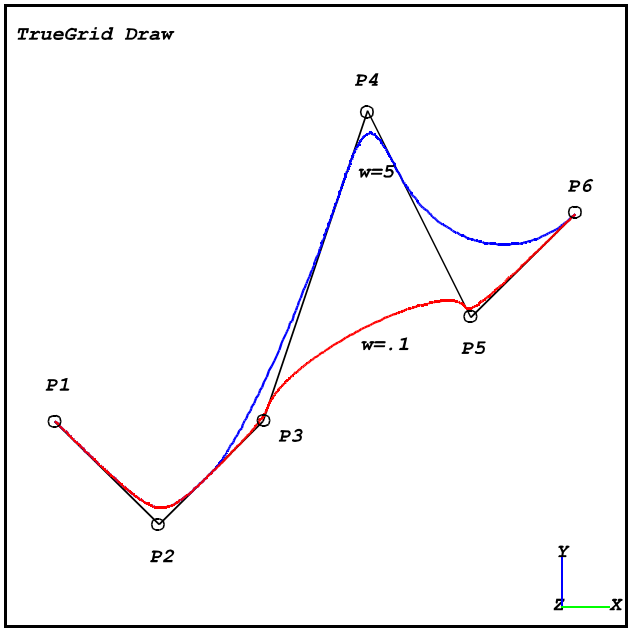


Figure 58 2 NURBS curves with various weights

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
c 6 control points
c P1 ...P6
c weight = 5 for P4 (x,y,z)

curd 3 nrb3
 0 0 0 1 2 4 5 5 5;  c knots
.1 1 1 .1 1 1;       c weights
     0 1 0           c control
     1 0 0           c points
     2 1 0
     3 4 0
     4 2 0
     5 3 0;;
c curve definition - 3
c nurbs
c 0 0 0 multiple knot
c 5 5 5 multiple knot
c 6 control points
c P1 ... P6
c weight = .1 for P4
c weight = 5 for P4
```

Example

```
curd 4 nrb3
0 0 0 1 1 2 4 5 5 5; c knots
.1 1 1 1 10 1 1;  c weights
     0 1 0   c control points
     0 .5 0
     1 0 0
     2 1 0
     3 4 0
     4 2 0
     5 3 0;;

c NURBS curve with
c discontinuity - multiple
c internal knots 1 1
c for P3
```

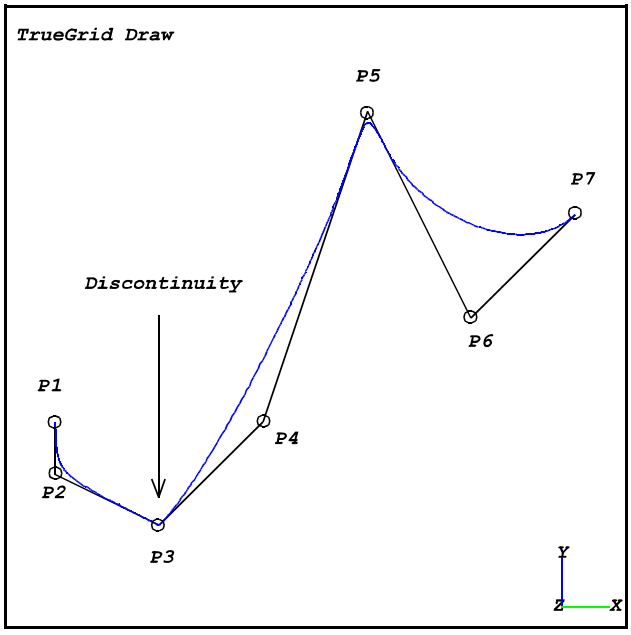


Figure 59 NURBS curve with discontinuity of the slope in P3

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

ld2d3d create/append a 2D curve converted to 3D

ld2d3d *2D_curve_# system coordinate start end transform ;*

where

system can be

rt for Cartesian coordinate interpolation,
cy for cylindrical coordinate interpolation, or
sp for spherical coordinate interpolation

coordinate can be

x for the first coordinate interpolated
y for the second coordinate interpolated
z for the third coordinate interpolated

start is the first value of the specified coordinate,
end is the last values of the specified coordinate, and
transform is an optional transformation that may be applied.

Remarks

The interpolation formula is: $x = (1 - \alpha)X_0 + \alpha X_1$

where x is the interpolated coordinate,

α is a parametric coordinate, it goes from 0 to 1, and

X_0, X_1 are the coordinates of the starting and ending points.

The parametric coordinate is segmented, with the segmentation based on the arc length of the curve. See the examples on the following page.

A transformation is formed by a sequence of simple transformations such as rotate and translate. For a complete list of these simple transformations, see the **lct** command.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
ld 1 lep 2 1 0 0 -100 100 0 ;
c 2d curve definition
c elliptic arc

curd 1 ld2d3d 1 rt x 2 2 ;
c 3d curve - 1 definition
c elliptic arc
c in plane

curd 2 ld2d3d 1 rt x 2 -2 ;
c 3d curve - 2 definition
c elliptic arc
c with nonuniformly
c interpolated coordinate x
```

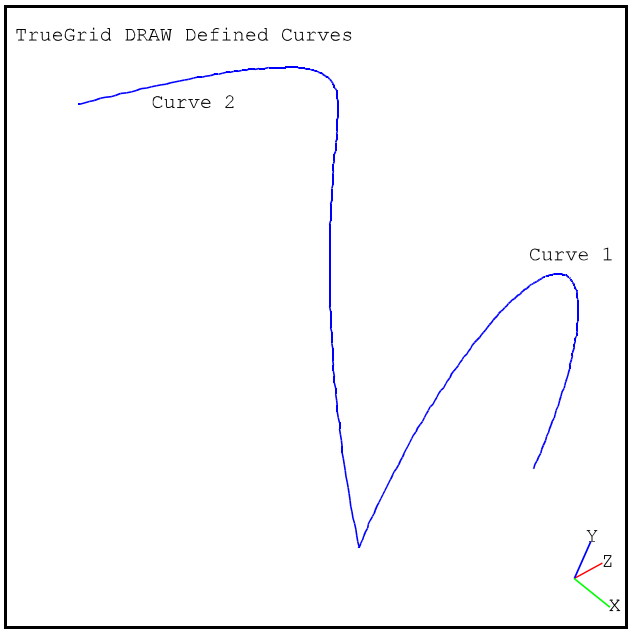


Figure 60 curve 1 and curve 2 by ld2d3d

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

intcur create/append a 3D curve by interpolation

intcur *3D_curve_1 3D_curve_2 interpolation_parameter transform ;*

where

3D_curve_1	is the ID number of a 3D curve,
3D_curve_2	is the ID number of the second 3D curve,
interpolation_parameter	is the distance from curve_1 (towards the second), and
transform	is a final transformation, which may be applied.

Remarks

The **intcur** command is especially useful for defining the inside curves for a butterfly mesh which must be applied along curved geometry.

A transformation is formed by a sequence of simple transformations such as rotate and translate. For a complete list of these simple transformations, see the **lct** command.

Example

Curve **4** is created by interpolation between curves **2** and **3** with interpolation parameter .3 (**Figure 61**). Simplified command file:

```
curd 4 intcur 2 3 .3  ;
c definition of curve 4 by
c interpolation
c between curves 2 and 3
```

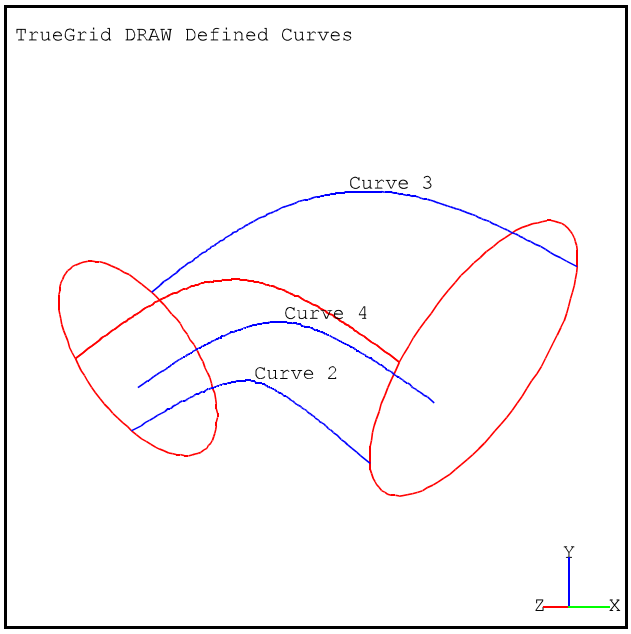


Figure 61 curve 4 by intcur

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

lp3pt create/append a 3D curve by pairs of defined points

lp3pt *point_id_1 ... point_id_n ; trans ;*

Remarks

The **lp3pt** command connects points on surface/curve by line segments. In the case of infinite surfaces (cylinder, plane, cone...), the surface must first be displayed. You can label these points by choosing ***Labels*** and ***Surf Point***. If you do not see a label where you expect one, then zoom in a bit further. If you click on a label you can then enter that label (or point ID) using the **F8** function key.

A transformation is formed by a sequence of simple transformations such as rotate and translate. For a complete list of these simple transformations, see the **lct** command.

Example

```
sd 1 function -90 90 -90 90
   cos(u); cos(v);cos(u)  ;;
curd 1
lp3pt 1.50.3 1.40.6 1.30.50
     1.24.54 1.2.1  ;  ;
     c curve 1 is created
     c from segments
     c connecting points
     c 1.2.1  1.24.54  1.30.50
     c 1.40.6 1.50.3
```

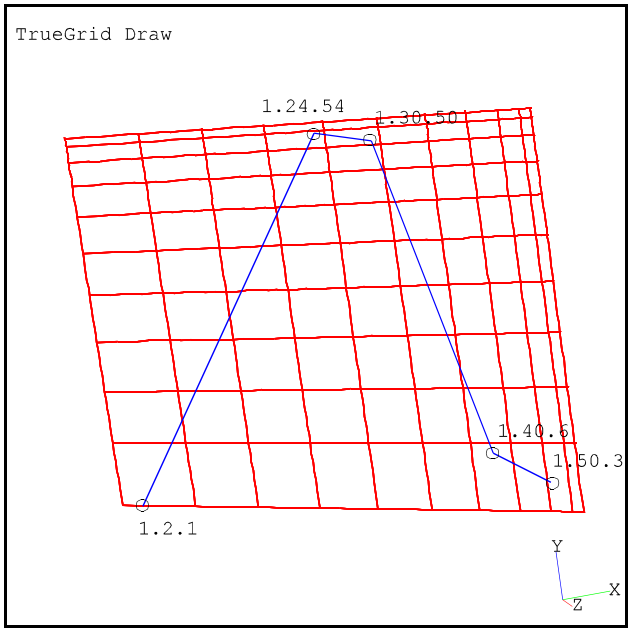


Figure 62 curve by lp3pt

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

3dfunc create/append a parameterized function curve

3dfunc *min_u max_u x_expres ; y-expres ; z-expres ; trans ;*

where

min_u	is the minimum value of the parameter, u,
max_u	is the maximum value of the parameter,
x_expres	is the function of u which defines the x-coordinates,
y-expres	is the function of u which defines the y-coordinates,
z-expres	is the function of u which defines the z-coordinates, and
trans	is an optional final transformation.

Remarks

This command enables you to define a parameterized 3D curve over a specified domain.

A transformation is formed by a sequence of simple transformations such as rotate and translate. For a complete list of these simple transformations, see the **lct** command.

Example

```
curd 1 3dfunc -90 90 cos(u);
sin(u) ; sin(u)*cos(u) ; ;
c u goes from -90 to 90
c x = cos(u)
    y = sin(u)
    z = sin(u)*cos(u)
```

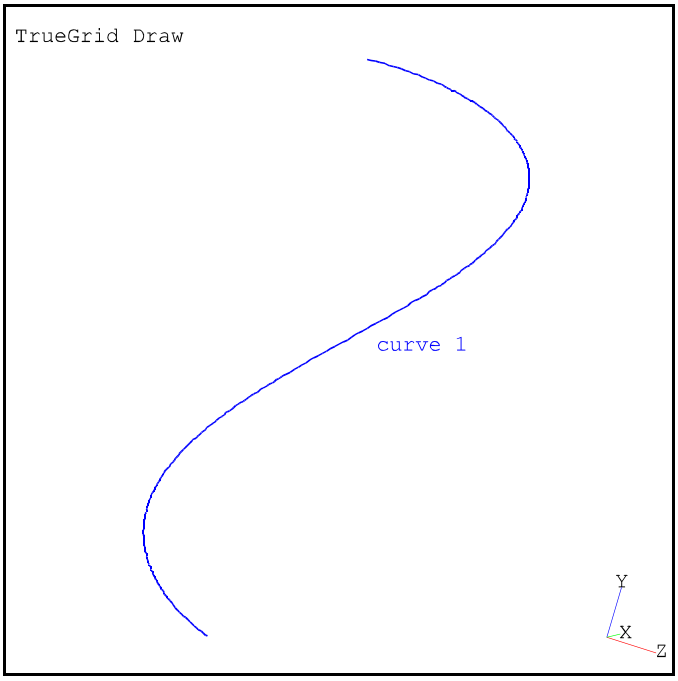


Figure 63 curve by 3dfunc

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

projcur create/append curve projected onto a surface

projcur *3D_curve_# surface_# trans ;*

where

3D_curve_#	is the ID number of a previously-defined 3D curve,
surface_#	is the ID number of a previously-defined surface, and
trans	is an optional transformation.

Remarks

The **projcur** command creates a curve by projecting an existing curve onto a surface. Projection is sensitive to surface curvature and/or the relative position of the curve to the surface. Also, see the following **pscur** command, which is similar to this command except that it also smooths the projected curve.

A transformation is formed by a sequence of simple transformations such as rotate and translate. For a complete list of these simple transformations, see the **lct** command.

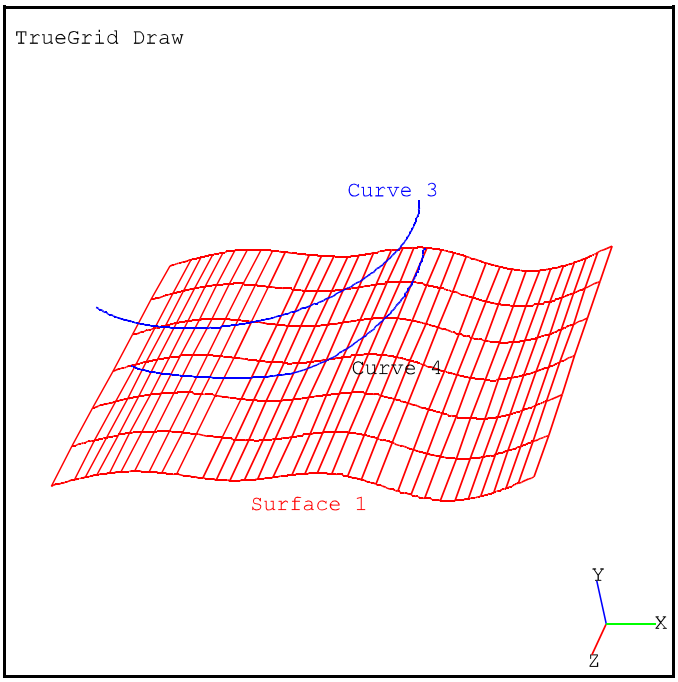


Figure 64 curve by **projcur**

Example

```
curd 4 projcur 3 1 ;
c curve 4 by projection of
```

pscur create/append curve projected onto a surface and smoothed

pscur *3D_curve_# surface_# #_iterations tol trans ;*

where

3D_curve_#	is the ID number of an existing 3D curve,
surface_#	is the ID number of an existing surface,
#_iterations	is the number of iterations used,
tol	is the tolerance which limits the smoothing iterations, and
trans	is an optional final transformation.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

This command projects the specified 3D curve onto the specified surface and then iteratively smooths the final curve. This iterative method continues until the differences between successive iterations are less than *tol*.

A transformation is formed by a sequence of simple transformations such as rotate and translate. For a complete list of these simple transformations, see the **lct** command.

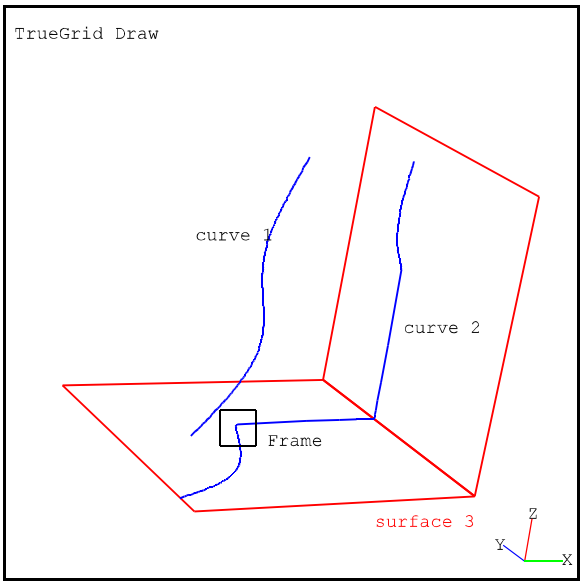


Figure 65 curve2 by pscur

Example

The **pscur** command is demonstrated by projection and smoothing of the spline curve onto discontinuous planar surface (**Figure 65**). **Figure 66** is created from **Figure 65** by magnification of the Frame. It shows the difference between the **pscur** and **projcur** command when applied to the same curve. **pscur** creates the smoother curve.

```
curd 2 pscur 1 3 100 .01  ;

     c curve 2 - by projection
     c and smoothing of
     c curve 1 onto surface 3
     c with maximum
     c 100 iterations
     c and with minimum
     c tolerance .1
```

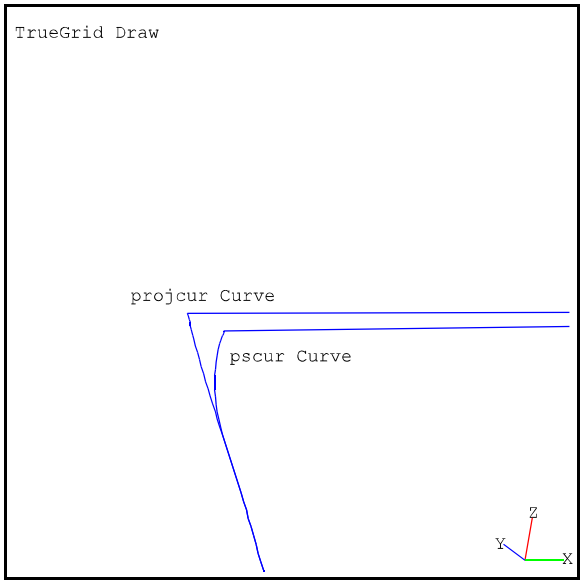


Figure 66 pscur vs projcur curve

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

arc3 create/append arc of a circle

arc3 *option system point system point system point*

where

option can be

seqnc to specify the arc starting at the first point, passing through the second, and ending at the third point.

cmplt to specify the complementary arc of the circle specified in seqnc, or

whole to specify the entire circle, and

system and *point* can be

rt *x y z* for cartesian coordinates,

sp *rho theta phi* for spherical coordinates,

cy *rho theta z* for cylindrical coordinates,

pt *surface.i.j* for a point off of a surface, or

pt *curve.i* for a point off of a 3D curve.

Remarks

This command appends a 3D arc of a circle passing through 3 points. There are three options. An arc can be created passing from the first to the second to the third point - **seqnc** option (**Figure 67**). The second option - **cmplt** is the complement of the first option above (**Figure 68**). The third option - **whole** creates the entire circle (**Figure 69**).

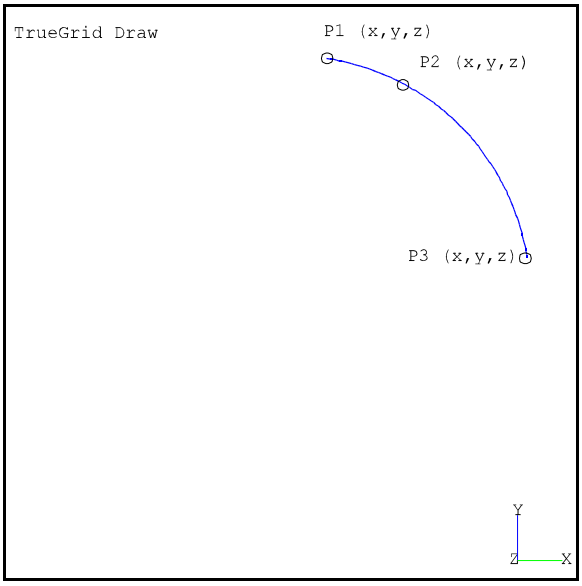


Figure 67 curve by arc3 seqnc

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
curd 1 arc3  seqnc  rt 0 1 0  rt .5 .8 0  rt 1 0 0 ;
c  circular arc - seqnc
c  given by points
c  P1(0,1,0)
c  P2(.5,.8,0)
c  P3(1,0,0)
```

```
curd 1 arc3
cmplt rt 0 1 0
rt .5 .8 0  rt 1 0 0 ;
c  circular arc - cmplt
c  given by points
c  P1(0,1,0)
c  P2(.5,.8,0)
c  P3(1,0,0)
```

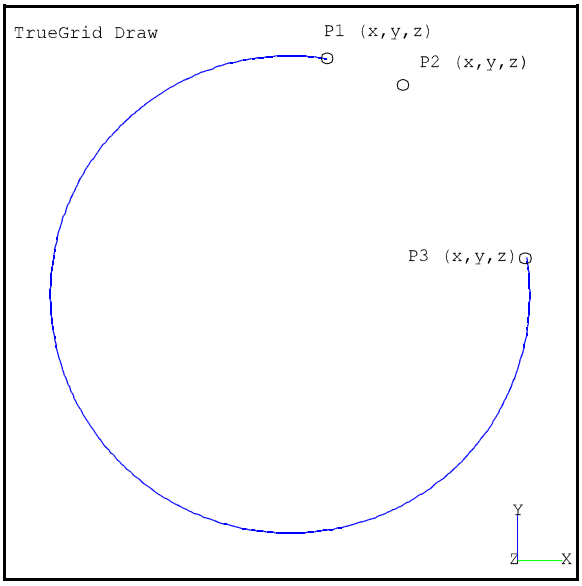


Figure 68 curve by arc3 cmplt

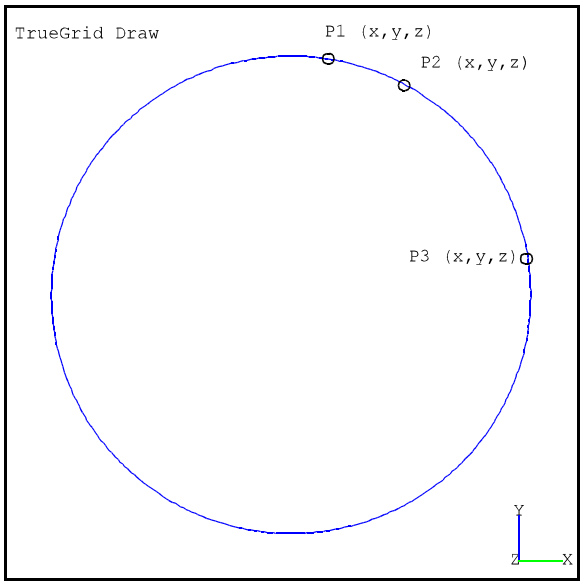


Figure 69 curve by arc3 whole

```
curd 1 arc3 seqnc rt 0 1 0
rt .5 .8 0  rt 1 0 0 ;
c  circle - whole
c  given by points
c  P1(0,1,0)
c  P2(.5,.8,0)
c  P3(1,0,0)
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

cpcd create/append a copy of a previously defined 3D curve.

cpcd *curve_# trans* ;

where

curve_# is the ID number of a previously-defined 3D curve, and

trans is an optional final transformation.

Remarks

This option allows you to copy a previously-defined 3D curve and transform it.

A transformation is formed by a sequence of simple transformations such as rotate and translate. For a complete list of these simple transformations, see the **lct** command.

Example

```
c definition of spline curve
curd 1 csp3 loop
     0  -1   0
     3  -1.5 0
     5  -1   0
     5   0   0
     5.1 1   0
     3   .5  0
     0   1   0; ;
curd 2 cpcd 1 mz 5 csca .5 ;
c definition of curve 2
c  by copy from curve 1
c and moved in the z-direction
c and scaled to the half size
```

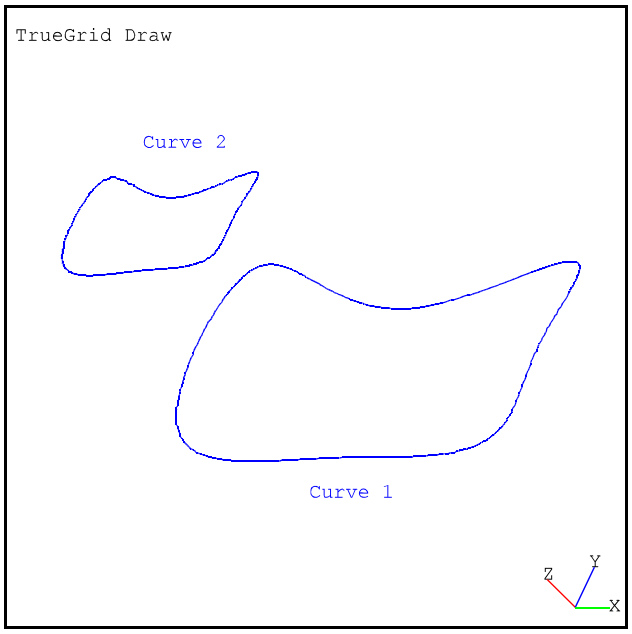


Figure 70 curve 2 by **cpcd** from curve 1

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

cpcds create/append a copy of previously defined 3D curves

cpcds *list_curve_# trans* ;

where

list_curve_# is an ordered list of 3D curve ID numbers, and

trans is an optional final transformation.

Remarks

The *list_curve_#* is ordered (**Figure 71**).

A transformation is formed by a sequence of simple transformations such as rotate and translate. For a complete list of these simple transformations, see the **lct** command.

Example

```
curd 5 cpcds 1 2 3 4;
rzx       c symetry - zx plane
my -1     c translation - y
mz .5     c translation - z
rz 10 ;   c rotation ar.- z

c definition of curve 5
c by copy of curves 1,2,3,4,5
```

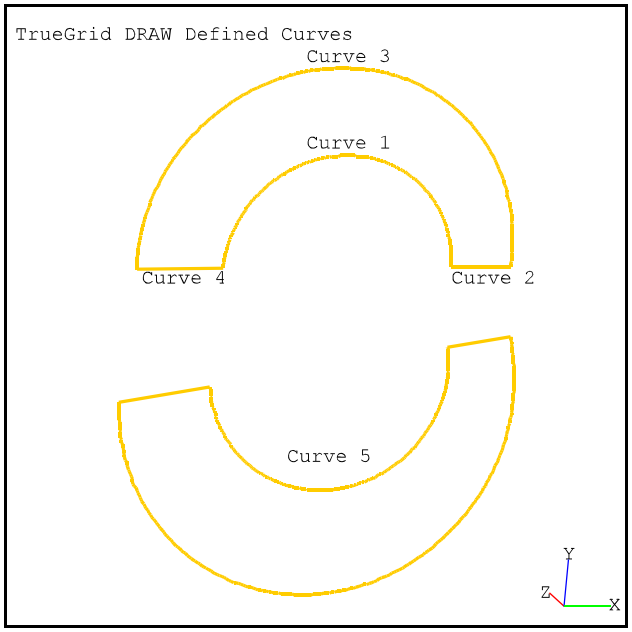


Figure 71 curve 5 by cpds

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

twsurf create/append the curve at the intersection of two surfaces

twsurf *surface_1 surface_2 x1 y1 z1 ... xn yn zn ; trans ;*

where

surface_1	is the ID number of the first surface,
surface_2	is the ID number of the second surface,
x1	is the approximate x-coordinate of the curve's beginning,
y1	is the approximate y-coordinate of the curve's beginning,
z1	is the approximate z-coordinate of the curve's beginning,
xn	is the approximate x-coordinate of the curve's end,
yn	is the approximate y-coordinate of the curve's end,
zn	is the approximate z-coordinate of the curve's end, and
trans	is an optional final transformation.

Remarks

This command creates/appends the curve at the intersection of two surfaces. Initial points are required to determine the endpoints and some intermediate points for surfaces with large curvature (greater than 120 degrees). These points need only be approximate.

A transformation is formed by a sequence of simple transformations such as rotate and translate. For a complete list of these simple transformations, see the **lct** command.

Example

A curve is created at the intersection of a torus and a cylinder. The initial points are picked by ***Projection*** on Surface 1 (**Figure 72**). The points form an oriented curve, which is used as a first approximation of intersection. The curve is closed by coincident starting and ending initial points (**Figure 73**). The command file follows:

```
sd 1 ts 0 0 0 0 0 1 1 0 .5
     c definition of torus
     c - surface 1

sd 2 cy .9 .9 0 0 .0 1 .4
     c definition of
```

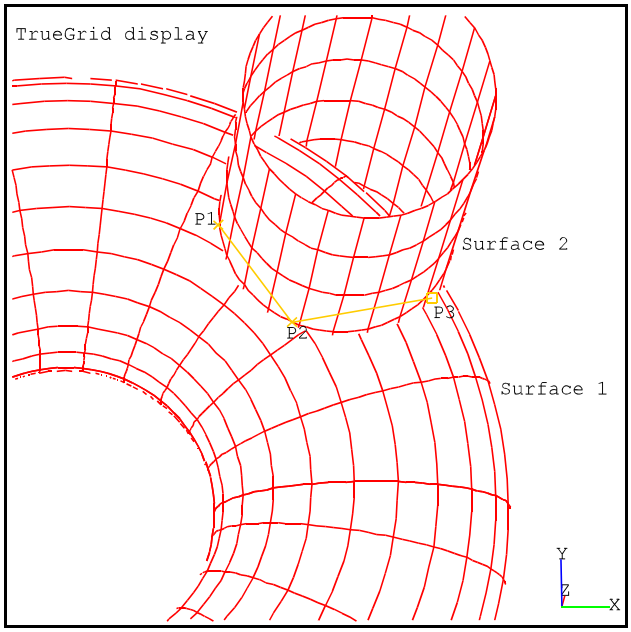


Figure 72 First 3 initial points of Curve 1 picked by Projection on Surface 1

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
     c cylinder - surface 2

curd 1
twsurf 1 2
 5.47e-01 1.06e+00  4.60e-01
 4.99e-01 8.72e-01  4.99e-01
 7.70e-01 5.31e-01  4.95e-01
 1.20e+00 6.59e-01  3.29e-01
 1.26e+00 7.52e-01 -1.74e-01
 9.79e-01 4.96e-01 -4.90e-01
 5.77e-01 6.65e-01 -4.85e-01
 5.48e-01 1.09e+00 -4.47e-01
 7.76e-01 1.28e+00  4.41e-02
 5.47e-01 1.06e+00 4.60e-01;;;

     c definition of curve 1
     c at the intersection of
     c surface 1 and 2 with 9
     c initial points
     c (coincident starting
     c and ending initial
     c point)
```

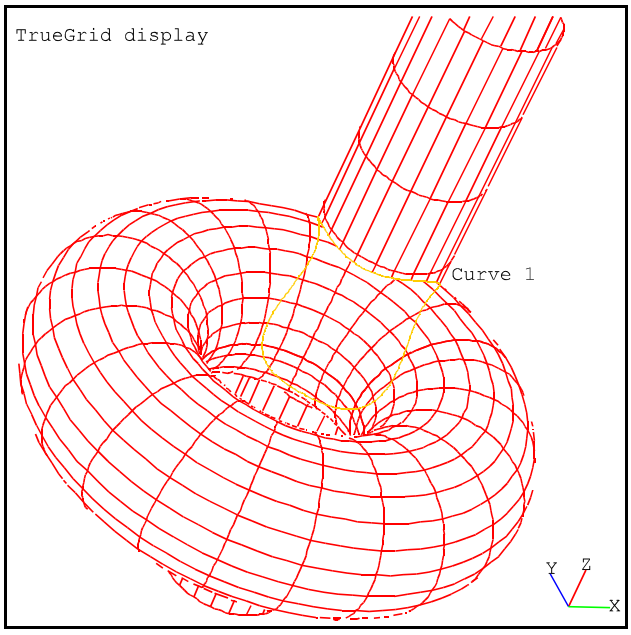


Figure 73 Curve by **twsurf**

5. Display 3D Curve

dcd display a 3D curve

dcd *3d_curve_#*

dcds display a set of 3D curves from a list

dcds *3d_curve_list* ;

dacd display all 3D curves

dacd (no arguments)

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

acd add a 3D curve to the picture

acd *3d_curve_#*

acds add a list of 3D curves to the picture

acds *3d_curve_list* ;

rcd remove a 3D curve from the picture

rcd *3d_curve_#*

rcds remove a list of 3D curves from the picture

rcds *3d_curve_list* ;

racd remove all 3D curves from the picture

racd (no arguments)

lacd list all of the active 3D curves in the picture

lacd (no arguments)

6. Print 3D Curves

cdinfo print information about the 3D curves

cdinfo *<no arguments>*

Remark

Prints the number of points and segments of each 3D curve.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
igescd 42 45 3;      c evaluate IGES curves 42-45
labels sdedge        c show the edge labels to the surfaces
curd 1
sdedge 2.2 sdedge 2.4
rmseg
sdedge 2.1 sdedge 2.4
cdinfo c write information about the 3D curves
```

The **cdinfo** command writes the following table:

```
        3D CURVE INFORMATION
         curve      1 has     71 points and      3 segments
         curve      3 has    257 points and      1 segments
         curve      4 has      2 points and      1 segments
         curve      5 has      2 points and      1 segments
         curve      6 has    188 points and      1 segments
```

7. Delete 3D Curve

rmseg remove last segment from 3D curve

rmseg <*no arguments*>

Remark

Remove the last segment appended to a 3D curve by **curd**.

delcd delete a 3D curve from the True*Grid*® data base

delcd *3d_curve_#*

Remarks

The deletion of a 3D curve cannot be undone. If you wish to remove a 3D curve from the picture, use **rcd**.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

delcds delete a list of 3D curves from the True*Grid*® data base

delcds *3d_curve_list* ;

Remarks

The deletion of a 3D curve cannot be undone. If you wish to remove a list of 3D curves from the picture, use **rcds**.

8. Surfaces

The following is a full description of all available methods for creating a surface. Commands that require a surface, such as the **sf** and **sfi** commands refer to a surface defined using the **sd**, **vpsd**, **iges**, **igessd**, **igespd**, or **nurbs** commands. Of this group of surface definition commands, only the **sd** command can specify the shape of the surface. The other surface definition commands import surfaces from a file and use the shape of the surface described in the file. Surfaces defined with the **sd** command have the advantage of, in many cases, of being simple, and in all cases can be formed using parameters and expressions so that you can have any degree of parameterization you desire. The disadvantage is that the parametric form of the surfaces can be less intuitive and require a more analytic approach to designing the desired shape.

When a surface is defined, it must be tessellated to be drawn. This tessellation is a set of connected points forming linear polygons as an approximation to the surface. It is these polygons that appear in the drawing. In many cases, they look like a block mesh. These tessellation points are assigned labels of the form s.i.j where s is the surface number, i is the first index of the point in the grid of tessellation, and j is the second index. In a more complex surface with trimming curves or in a polygon surface, there are usually triangles. The same notation is used but with a different meaning. The first index is the polygon number. The second index is the sequence number of the point used

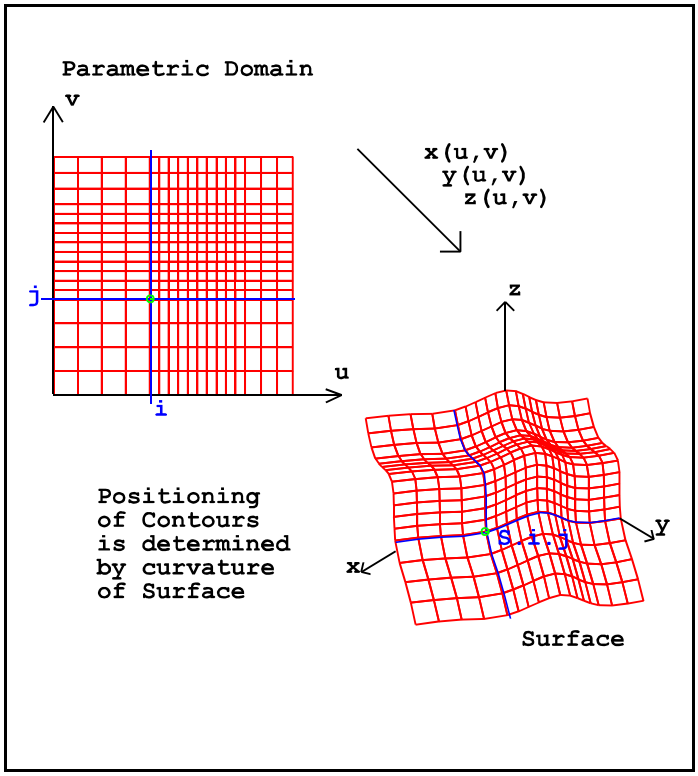


Figure 74 Tessellation

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

to define the polygon.

In some surface types (**csps**, **bsps**, **nrbs**, and **hermite**), you can specify boundary conditions (i.e. the derivatives along the edges). The easiest thing to do is to select the natural boundary condition. This means that the 2nd derivatives are set to zero. If you choose to specify the derivatives along an edge of the surface, you must specify the derivative for each control point along that edge. For example, suppose you choose to build a cubic spline surface with 6 control points in each row (u-direction) and 3 control points in each column (v-direction) of the table of control points. The edges are numbered. The edge corresponding to the minimum u is edge 1, maximum u is 2, minimum v is 3, and maximum v is 4. If you chose to specify derivatives along edge 1 or 2, you will need 3 derivatives. If you chose to specify derivatives along edge 3 or 4, you must specify 6 derivatives. Each derivative is a vector with three components forming the direction vector for the derivative. A flag formed by 4 digits are used to flag the type of boundary condition for each of the corresponding edges. 0 means a natural derivative along the edge and 1 means that the derivatives are being specified. All derivatives follow this 4 digit flag, in order. In some cases, you can specify a loop by setting either the first 2 digits in the flag or the last 2 digits to 2.

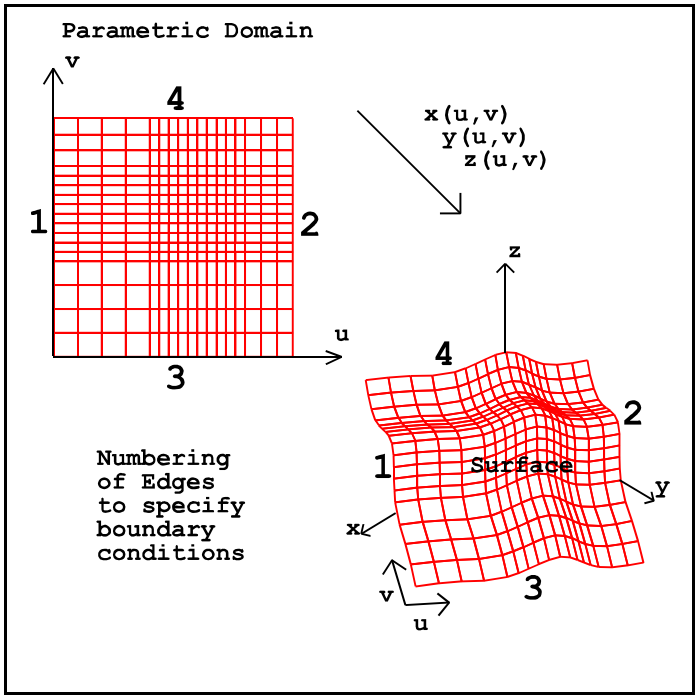


Figure 75 Numbered edges

In the case of infinite surfaces (**plan**, **xyplan**, **yzplan**, **zxplan**, **pl3**, **pl3o**, **iplan**, **cy**, **cyr2**, **cyr3**, **xcy**, **ycy**, **zcy**, **cp**, **cn2p**, **pr**), the surface tessellation depend on the size of the bounding box for the graphics. The size of the bounding box is determined from the size of the finite surfaces in the model. In an infinite surface, the points used to tessellate the surface are recalculated after any change of the bounding box. The bounding box is recalculated with the **restore** command. The bounding box can be displayed by the **grid** command.

If an infinite surface is not drawn, which is usually the case when a command file is run in batch mode, then the points of tessellation will not be calculated. When you select points or edges of a surface symbolically (by label), be sure that surface has been drawn. When these points are selected

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

by label, the coordinates are substituted in the save file, not its label.

accuracy set accuracy of surface projections

accuracy *acc*
where *acc* is the scale factor relative to the default accuracy of the projection

Remarks

The default accuracy parameter is 1.0 which gives you 3.5 digits of accuracy. It must be positive. An accuracy of 2 or more will give you double precision projections.

chkfolds check for folds in surfaces

chkfolds *option*
where the *option* can be

on	activate the fold checking projection algorithm
off	deactivate the fold checking projection algorithm (default)

Remarks

There is a problem in accuracy when projecting to a surface that has folds in it. To avoid additional costly calculations for all surface projections, this problem is corrected by issuing a flag only when a surface is folded and the projection is not done correctly. This flag is set using the **chkfolds** command. This is a very rare condition. It effects all projections within a part.

delsd delete a surface

delsd *surface_#*
where *surface_#* is the assigned surface identification number

Remarks

When a surface is deleted, it cannot be undone. Do not delete a surface that is being used by a part. Wait until the part is complete before deleting the surface. Actually, this is a useless command since one can avoid a surface by simply not showing it.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

delsds delete a set of surfaces

delsds surface_list ;
where a *surface_#* is the assigned surface identification number

Remarks

When a surface is deleted, it cannot be undone. Do not delete a surface that is being used by a part. Wait until the part is complete before deleting the surface. Actually, this is a useless command since one can avoid a surface by simply not showing it.

fetol feature extraction from polygon surfaces

fetol *angle*
where *angle* is the maximum angle for a feature

Remarks

Interior edges are formed within a polygon surface when the angle between two facets or polygons fall below the specified maximum angle. The polygon surfaces affected by this command come from the **vpsd** command and the **sd** with the **mesh**, **face**, **faceset**, **poly**, **stl**, or **bstl** options. These interior edges are useful when attaching edges of the mesh to important features of the polygon surface.

The **fetol** command must precede the creation of the surface.

The default is no interior edges. To change **fetol** back to the default, use and angle greater than 180 degrees.

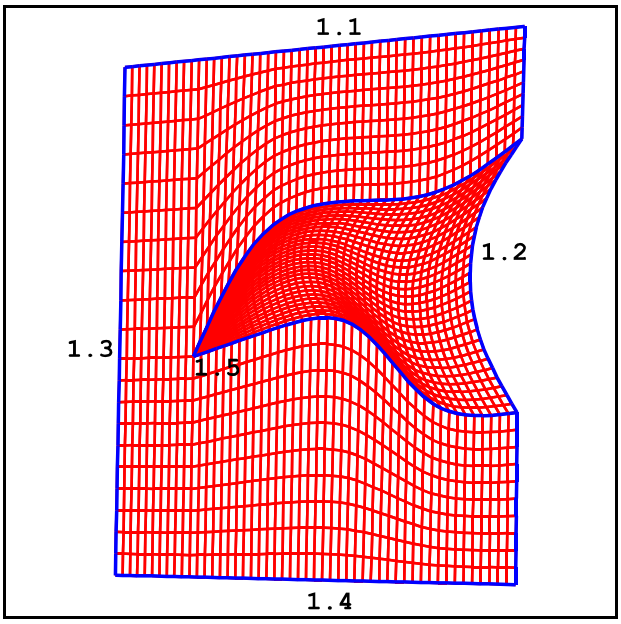


Figure 76 Interior feature edge 1.5

Example

```
fetol 163
vpsd 1 bed.cor bed.elm;
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

getol relative tolerance to tessellate curves and surfaces

getol *tolerance*
where *tolerance* is the relative tolerance

Remarks

The default relative tolerance for geometry is 1.0. This geometry tolerance is a factor used in the tessellation process. The actual tolerance is based on the total size of the geometry entity. Areas of greater curvature result in more points/polygons in the tessellation. When **getol** is increased, the number of points/polygons is decreased. Decrease **getol** to increase the number of points/polygons. This will also affect the amount of memory consumed for graphics and in some cases affect the accuracy of the projection when the **accuracy** parameter is set below 2.

lcsd list all composite surfaces with a specified surface

lcsd *surface_#*
where *surface_#* is a valid surface number

Remarks

This function helps identify a specific composite surface when there are many.

mvpn modify a surface node of a polygon surface

mvpn *point_id coor_id change(s)*
where

point_id	identify the surface node which has the form *s.p.n*
where	
s	surface number
p	i-index or polygon number of the surface
n	j-index or node number of the polygon
coor_id	identify the coordinates to be changed
can be	
x	change the x-coordinate
y	change the y-coordinate
z	change the z-coordinate

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

xy	change the xy-coordinates
yz	change the yz-coordinates
xz	change the xz-coordinates
xyz	change the xyz-coordinates
change(s)	list of coordinate changes corresponding to *coor_id*.

Remarks

This command is used to move polygon surface nodes of types **faceset**, **poly**, **intp**, **stl**, and **bstl**. Polygon surfaces can be crude and error prone approximations to the actual surfaces, and as such, may produce polygon surfaces that are absurd to mesh. This feature can help you smooth the polygon surface to make the problem of meshing practical to solve. If you wish, you can save the modified polygon surface by using the **wrsd** command so that the next time you use this polygon surface, you do not have to repeat these polygon surface modifications.

Use the labels command to label the surface nodes on a polygon surface to aid in selecting the *point_id*.

npll set the quality of the projection to surfaces

npll *size*

where *size* is a relative curvature in a fold in the surface

Remarks

This command is almost never needed. In fact, there has been only one known case where this was needed. It is used in the special case that the surface being projected onto is extremely curved and almost folds onto itself. If the mesh does not project properly, this command will increase the quality of the projection search and perhaps fixes the problem. An alternative solution is to break the surface into smaller surfaces or use 3D curves with additional partitions in the mesh to constrain the mesh to the desired region of the surface. The default for this parameter is 30. It cannot be larger than 2000. If this number is increased, the length of time needed to project onto a surface will be increased. The cost for the projections should increase by something less than linear with respect to the number assigned with the command **npll**. If you are having difficulties, also try the **getol** command. This command is not found in the Graphical User Interface because it should only be used by experts who know to use it very sparingly.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

project project a point to surface(s)

project *x y z list_of_surface_numbers* ;

where

x y z	coordinates of the initial point
list_of_surface_numbers	list of surface numbers

Remarks

Project uses the given 3D point as the initial point for the projection algorithm. This point is projected to the closest point in the intersection of the given surfaces.

This function returns the coordinates of the point of projection and the normal vector to each of the surfaces at that point of projection. These values can be used in expressions by referencing the following parameters:

xprj - x-coordinate due to the **project** command
yprj - y-coordinate due to the **project** command
zprj - z-coordinate due to the **project** command
xnrm - x-component of the normal to the 1st surface from the **project** command
ynrm - y-component of the normal to the 1st surface from the **project** command
znrm - z-component of the normal to the 1st surface from the **project** command

This function is of little use in directly generating a mesh. Use the **sf** or the **sfi** command or click on the ***Project*** button to constrain (or project) a face of a mesh to a surface. The **project** command is intended to be used to demonstrate the projection method without the need to build a part. It can also be used to form parameters in creating geometry. For example, you can use the normal to define the tangent of a spline curve so that the curve is orthogonal to the surface. You can parameterize a normal offset from a surface, again by using the normal of the projection. You may find other creative uses for this function.

For example, if one surface is given, then the result of the function is the coordinates of the point on that surface which is closest to the original point.

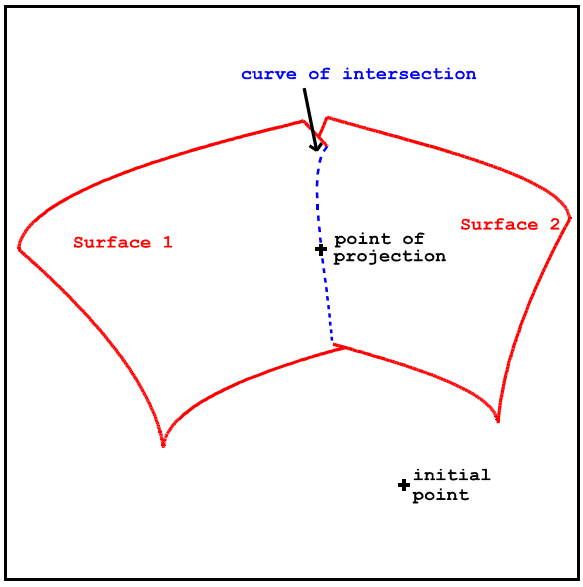


Figure 77 Project To Two Surfaces

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

If two surfaces are specified, then the result will be the point of projection along the intersecting curve of the two surfaces. If three surfaces are specified, then the result will be the point of intersection of the three surfaces.

Example

```
project 1.2 2.3 3.4 17 151;
```

This projects the point (1.2,2.3,3.4) to the intersection of surfaces 17 and 151. The results of this command are printed to the text window and the tsave file.

```
(x,y,z)=  -7.07107E-01  -3.33071E-08   7.07107E-01
 Table Of Distances To Surfaces
 distance to surface 17 is   0.00000E+00
 normal=-7.07107E-01 -3.33071E-08  7.07107E-01
 distance to surface 151 is   2.86341E-08
 normal= 5.77350E-01  5.77350E-01  5.77350E-01
```

pvpn place a surface node of a polygon surface

pvpn *point_id coor_id coordinate(s)*

where

point_id	identify the surface node which has the form *s.p.n*
where	
s	surface number
p	i-index or polygon number of the surface
n	j-index or node number of the polygon
coor_id	identify the coordinates to be changed
can be	
x	change the x-coordinate
y	change the y-coordinate
z	change the z-coordinate
xy	change the xy-coordinates
yz	change the yz-coordinates
xz	change the xz-coordinates
xyz	change the xyz-coordinates
coordinate(s)	list of coordinates corresponding to *coor_id*.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

This command is used to move polygon surface nodes of types **faceset**, **poly**, **intp**, **stl**, and **bstl**. Polygon surfaces can be crude and error prone approximations to the actual surfaces, and as such, may produce polygon surfaces that are absurd to mesh. This feature can help you smooth the polygon surface to make the problem of meshing practical to solve. If you wish, you can save the modified polygon surface by using the **wrsd** command so that the next time you use this polygon surface, you do not have to repeat these polygon surface modifications.

Use the labels command to label the surface nodes on a polygon surface to aid in selecting the *point_id*.

Alternatively, you can use the graphical user interface to move a surface node of a polygon surface. There are two basic ways to do this. In both cases, choose the surface node of the polygon surface to be moved by selecting the ***Poly Surface*** button in the ***Move Pts.*** panel of the environment window. Then move the mouse close to the surface node that you wish to move and click on the **F5** function key.

If you choose a surface node using the ***Pick*** panel, you must then click on the ***Attach*** button to cause the surface node to move. This will produce a **pvpn** command that will be written to the text window and the tsave file.

If you choose the ***Move Pts.*** panel, you can then move the surface node with a click and drag of the left mouse button. When you release the mouse button, a **pvpn** command is generated.

sd surface definition

sd *[name] surface_# surface_type surface_parameters*

where

name	optional name for the surface
surface_#	positive integer to identify the surface

surface_type and *surface_parameter* can be any of:

Derivatives of other surfaces (see also **trsd**):

sds $sd_1\ sd_2\ ...\ sd_n\ ;$	combine several numbered surfaces into one
intp $sd_1\ sd_2\ fraction$	interpolate a surface between two surfaces

Planes:

plan $x_0\ y_0\ z_0\ x_n\ y_n\ z_n$	infinite planar surface by a normal vector

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

xyplan *trans ;* transform an infinite x-y plane
yzplan *trans ;* transform an infinite y-z plane
zxplan *trans ;* transform an infinite z-x plane
pl2 *system point system point*
plane specified by two points
pl3 *system point system point system point*
plane specified by three points
pl3o *system point system point system point offset*
plane specified by 3 points and an offset
iplan *a b c d* plane defined by an implicit function

Spheres:

sp $x_0\ y_0\ z_0$ *radius* sphere

Cylinders:

cy $x_0\ y_0\ z_0\ x_n\ y_n\ z_n$ *radius* infinite cylindrical surface
xcy *radius trans ;* transform an infinite x-axis cylinder
ycy *radius trans ;* transform an infinite y-axis cylinder
zcy *radius trans ;* transform an infinite z-axis cylinder
cy3 $x_1\ y_1\ z_1\ x_2\ y_2\ z_2\ x_3\ y_3\ z_3$ cylinder by 3 points
cyr2 $x_1\ y_1\ z_1\ x_2\ y_2\ z_2$ *radius* cylinder by 2 points and radius
cyr3 $x_1\ y_1\ z_1\ x_2\ y_2\ z_2\ x_3\ y_3\ z_3$ *radius*
cylinder by 3 points and radius

Conics:

cn2p $x_0\ y_0\ z_0\ x_n\ y_n\ z_n\ r_1\ t_1\ r_2\ t_2$ infinite conical surface by two points
cone $x_0\ y_0\ z_0\ x_n\ y_n\ z_n\ r\ \theta$ infinite conical surface by angle and radius
er $x_0\ y_0\ z_0\ x_n\ y_n\ z_n\ r_1\ r_2$ ellipse revolved about its axis
pr $x_0\ y_0\ z_0\ x_n\ y_n\ z_n\ r_1\ t_1\ r_2\ t_2\ r_3\ t_3$
infinite parabola revolved about an axis

Torus:

ts $x_0\ y_0\ z_0\ x_n\ y_n\ z_n\ r_1\ t\ r_2$ torus

2D Curves, surfaces formed from:

crx *ln* planar 2D curve rotated about the x-axis
cry *ln* planar 2D curve rotated about the y-axis
crz *ln* planar 2D curve rotated about the z-axis
cr $x_0\ y_0\ z_0\ x_n\ y_n\ z_n$ *ln* planar 2D curve rotated about an arbitrary axis
cp *ln trans ;* planar 2D curve extruded infinitely
rule2d y_1 *ln*$_1$ y_2 *ln*$_2$ *trans ;* ruled surface between two planar curves

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

swept *ln_0 direction ln_1 α_1 ... ln_n α_n ; trans ;*
surface with planar cross-sections along a 2D curve

3D curves, surfaces formed from:

rule3d *3D-curve$_1$ 3D_curve$_2$ trans ;*
ruled surface between two 3D curves

crule3d *3D-curve$_1$ 3D_curve$_2$ trans ;*
cylindrical surface between two 3D curves

pipe *3d_curve radius$_1$ arc_length$_1$ radius$_2$ arc_length$_2$ ... ; trans ;*
sweep a pipe shape along an arbitrary 3D curve

blend3 *3d_curve$_1$ 3d_curve$_2$ 3d_curve$_3$ trans ;*
blend three bounding 3D curves to form a patch

blend4 *3d_curve$_1$ 3d_curve$_2$ 3d_curve$_3$ 3d_curve$_4$ trans ;*
blend four bounding 3D curves to form a patch

r3dc *x_0 y_0 z_0 x_n y_n z_n 3D_curve start_angle end_angle trans ;*
3D curve rotated about an arbitrary axis

IGES (see also **iges**, **igesfile**, **igessd**, **igespd**, and **nurbsd**):

nurbs *nurbs* — IGES NURBS (Non-Uniform Rational B-Spline)

igess *surface trans ;* — IGES parametric surface

igesp *plane trans ;* — IGES plane,

Algebraic surfaces:

function *u_{min} u_{max} v_{min} v_{max} x-expression ; y-expression ; z-expression ; trans ;*
surface by three algebraic expressions

csps *#_columns #_rows flag conditions x_{11} y_{11} z_{11} x_{12} y_{12} z_{12} ... trans ;*
cubic spline surface

bsps *i-degree j-degree iknot$_1$ ... iknot$_{k1}$; jknot$_1$ .... jknot$_{k2}$; xcontrol$_1$ ycontrol$_1$ zcontrol$_1$ ... xcontrol$_n$ ycontrol$_n$ zcontrol$_n$ trans ;*
B-spline surface

nrbs *i-degree j-degree iknot$_1$ ... iknot$_{k1}$; jknot$_1$ .... jknot$_{k2}$; weight$_1$ ... weight$_n$; xcontrol$_1$ ycontrol$_1$ zcontrol$_1$ ... xcontrol$_n$ ycontrol$_n$ zcontrol$_n$ trans ;*
NURBS surface

hermite *#_columns #_rows flag conditions x_{11} y_{11} z_{11} x_{12} y_{12} z_{12} ... trans ;*
2nd order, once differentiable, spline surface

Polygonal surfaces (see also **vpsd**, **wrsd**, **mvpn**, **pvpn**, and **fetol**):

mesh *m n x_{11} y_{11} z_{11} x_{21} y_{21} z_{21} ... x_{mn} y_{mn} z_{mn} trans ;* — tabular data

face *region* — face of the present part (Part phase only)

faceset *set_name* — face set

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

poly *polygon_set trans ;* polygon set
stl *file trans ;* read the standard ASCII STL file
bstl *file trans ;* read the standard binary STL file

trans is a product from left to right of the following operators

mx *x_offset*
my *y_offset*
mz *z_offset*
v *x_offset y_offset z_offset*
rx *theta*
ry *theta*
rz *theta*
raxis *angle x0 y0 z0 xn yn zn*
rxy
ryz
rzx
tf *origin x-axis y-axis*
where each of the arguments consist of a coordinate type followed by coordinate information:

rt *x y z* (Cartesian coordinates)
cy *rho theta z* (cylindrical coordinates)
sp *rho theta phi* (spherical coordinates)
pt *c.i* (label of a labeled point from a 3D curve)
pt *s.i.j* (label of a labeled point from a surface)

ftf *1st_origin 1st_x-axis 1st_y-axis 2nd_origin 2nd_x-axis 2nd_y-axis*
where each of the arguments consist of a coordinate type followed by coordinate information:

rt *x y z* (Cartesian coordinates)
cy *rho theta z* (cylindrical coordinates)
sp *rho theta phi* (spherical coordinates)
pt *c.i* (label of a labeled point from a 3D curve)
pt *s.i.j* (label of a labeled point from a surface)

inv to invert the present transformation
csca scale all coordinates
xsca scale the x-coordinates
ysca scale the y-coordinates
zsca scale the z-coordinates

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

See the following section called Surface Dictionary for a full description of each surface type.

The surface name is optional. It still needs to be numbered. All surfaces with the same name are automatically combine into a composite surface and is known by that name.

Names must have some alphabetic characters in them and no spaces. Other special characters like : and ; should also be avoided, although the implications of using none alphanumeric characters has not been explored. This feature is case insensitive. Only the first 8 characters of a name are used.

A surface can only have one name, so it can be conveniently included into only one composite surface by name. To include it into additional composite surfaces, use the surface's number in the **sd** command with the **sds** option.

All surfaces are defined in the Cartesian coordinate system regardless of the coordinate system used to define the mesh (**block** or **cylinder**).

Sd defines a surface to be used by other commands such as **sf**, **project**, **ms**, and **ssf**.

You can look at pictures of the surfaces you define, with such commands as **dsd**. In fact, surfaces defined by the **sd** command are the only surfaces which you can see displayed this way. Besides **dsd**, the relevant commands are **asd**, **rsd**, **dsds**, **asds, rsds, dasd**, and **rasd**. When a surface is defined, the surface is evaluated at many points so that the surface can be rendered graphically. Areas of the surface which have greater curvature will be evaluated at more points. This is done once when the surface is defined. Some surfaces, such as a plane, cylinder, rotated parabola, cone, or extruded 2D curve, are infinite and can not be completely rendered. They are cut off in order to fit into the picture with the rest of the surfaces. When such a surface is added to the picture, it may extend a little further than desired. A restore of the picture (***Rest***) will re-evaluate the infinite surfaces so that they are extended uniformly.

You can import surfaces from a CAD/CAM system. These surfaces are assigned surface numbers using the **iges**, **igessd**, **nurbsd**, and **igespd** commands or with the **igess**, **nurbs**, or **igesp** options of the **sd** command. Surfaces can also be selected by their level number inherited from the IGES interface for viewing using the **lv**, **alv**, **rlv**, and **dlvs** commands. Associativity groups may also be used through definitions found in the IGES file to view these surfaces using the **dgrp**, **agrp**, **rgrp**, and **dgrps** commands

Defined surfaces can also be used to construct 3D curves by extracting an edge or contour of a surface using the **sdedge** and **contour** options of the **curd** command, respectively.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

smgap small surface gap tolerance

smgap *tolerance*
where *tolerance* is n absolute (positive) distance

Remarks

The **smgap** command sets a tolerance to small gaps in surfaces. If a surface gap is found to be smaller than the value of **smgap**, the gap is removed. This command is almost very needed because the default works in about %99.9999 of the time. This may be useful only in the case when a mapped surface has an (almost) degenerate edge. If this command is used before the command that creates the surface is issued, then the degenerate edge will not appear in the surface.

trsd transform a surface definition

trsd *surface_id trans* ;
where
trans can be any of:

mx *x_offset*	to translate along the x-axis
my *y_offset*	to translate along the y-axis
mz *z_offset*	to translate along the z-axis
v *x_offset y_offset z_offset*	to translate in a given vector direction
rx *theta*	to rotate about the x-axis
ry *theta*	to rotate about the y-axis
rz *theta*	to rotate about the z-axis
raxis *angle x0 y0 z0 xn yn zn*	to rotate about a general axis
rxy	to reflect about the xy-plane
ryz	to reflect about the yz-plane
rzx	to reflect about the zx-plane
tf *origin x-axis y-axis*	to transform to a frame of reference
ftf *1st_origin 1st_x-axis 1st_y-axis 2nd_origin 2nd_x-axis 2nd_y-axis*	to transform from one frame of reference to another
inv	to invert the transformation.

Each argument in the **tf** and **ftf** transformations consists of a coordinate type followed by coordinate information:

rt *x y z*	Cartesian coordinated
cy *rho theta z*	cylindrical coordinates

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

sp *rho theta phi*	spherical coordinates
pt *c.i*	label of a labeled point from a 3D curve
pt *s.i.j*	label of a labeled point from surface

Remarks

The *surface_id* can be its number or name.

This command applies a transformation to the given surface. The new surface assumes the surface *surface_#id* and the pre-transformed surface is lost. This is significant in the case of a combined surface (sds) because every surface in the list is replaced by its transformed version.

The transform is order dependent. Note that in the example (**Figure 78**), applying the shift before the rotation is different from applying the rotation before the shift.

Example

```
c Set up first surface
sd  3 function
    0 90 0 360
    cos(u)*cos(v);
    cos(u)*sin(v);
    3 *sin(u);;
sd  4 function
   -90 0 0 360
    cos(u)*cos(v);
    cos(u)*sin(v);
    .5*sin(u);;
sd  1 sds  3  4;
c Set up second surface
c (This copy is needed
c  because each surface
c  in the surface list
c  is transformed)
sd 30 function
    0 90 0 360
    cos(u)*cos(v);
    cos(u)*sin(v);
    3 *sin(u);;
sd 40 function
   -90 0 0 360
    cos(u)*cos(v);
    cos(u)*sin(v);.
    5*sin(u);;
sd 10 sds 30 40;
c Transform first surface by moving then rotating
```

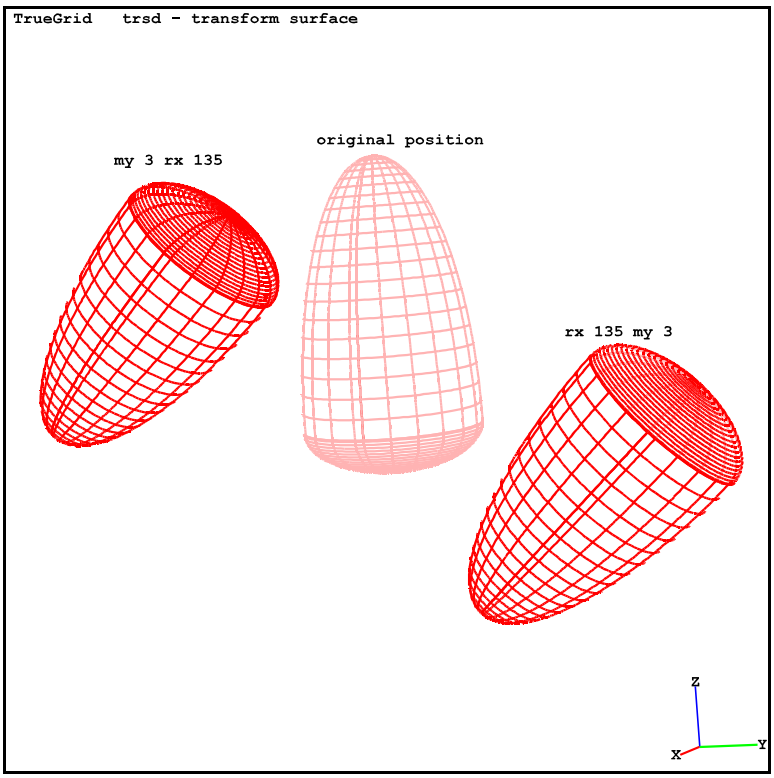


Figure 78: trsd - transforming a surface is order dependent

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
trsd 1 my 3 rx 135;
c Transform second surface by rotating then moving
trsd 10 rx 135 my 3;
```

sdinfo list surfaces

sdinfo *(no arguments)*

Use this command to see a list of all the numbered surfaces. For each surface number, **True*Grid*®** gives a terse description of the surface.

vd define a volume

vd *number type parameters*

where

number is a unique volume number with which you can refer to the volume later, and

type and *parameters* can be any of the following:

sp *x y z radius* for a sphere with center *x y z* and radius *radius*

cy *x y z xn yn zn radius* for an infinite cylinder of radius *radius* whose axis passes through *x y z* in direction (*xn*,*yn*,*zn*)

cr *x y z xn yn zn curve* for a rotated 2D curve. The curve number is *curve*, and the axis of rotation passes through *x y z* in direction (*xn*,*yn*,*zn*).

cyf *x y z xn yn zn radius z_min z_max* for a finite cylinder

sd *surface distance* for a surface with thickness *distance*. *surface* is the surface number.

box *xm ym zm xx yx zx option* ; for a box with corners at (*xm,ym,zm*) and (*xx,yx,zx*), also for LS-DYNA BOX

where *option* can be

adaptive *material level* for LS-DYNA BOX_ADAPTIVE

coarsen *inout_flag* for LS-DYNA BOX_COARSEN

Remarks

There are several uses of a volume. The **mtv** command uses a volume to modify the materials of elements within the volume. The Ls-dyna output uses the box volume to write *DEFINE_BOX cards.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

9. Surface Dictionary

The following is a list of surface types and parameters available, as options, under the **sd** command.

Many of the following surfaces refer to an axis to form a local coordinate system. The parameters $(x_0\ y_0\ z_0)$ give a point through which the axis passes. The parameters $(x_n\ y_n\ z_n)$ give the direction of the axis; that is, it is in the same direction as the line from the origin to the point $(x_n\ y_n\ z_n)$. The distance to this point must be positive. Often they also refer to *r* and *t*; *r* is a distance from the axis and *t* is the distance along the axis from the point $(x_0\ y_0\ z_0)$. Thus (*r*,*t*) form a local coordinate system in which we may define points, curves, etc..

Some of the following surfaces are based on a planar 2D curve defined with the **ld** command. They are assigned a number which is referred to by *ln*. The 2D curve must be defined first before it is referenced by the **sd** command.

Some of the following surfaces are based on a 3D curve defined with the **curd** command. They are assigned a number which is referred to by *3D_curve*. The 3D curve must be defined first before it is referenced by the **sd** command.

In some surface definitions, you make the surface in a convenient coordinate system and then immediately rotate or translate it to where you need it. This is indicated by the *trans ;* option in the surface description below. If you don't want any transformations, then you should simply omit any transformations and end the command with a semicolon.

blend3 blend three bounding 3D curves to form a patch

blend3 $3d_curve_1\ 3d_curve_2\ 3d_curve_3$ *trans ;*

where

$3d_curve_i$	3D curve number
trans	sequence of simple operators to transform the coordinates

Remarks

The three 3D curves used to form the boundaries of a surface patch are assumed to intersect. Care must be taken to insure that the 3D curves almost intersect. The algorithm can deal with gaps between the 3D curves, but the result can be unsatisfactory if the gaps are very large. The curves do not need to end at the desired corner points of the surface patch.

It is also assumed that the three curves surround only one 3-sided region. If, due to curvature, there

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

is an ambiguity as to which enclosed region will be surfaced, the results may be unsatisfactory. In such a case, we recommend that you break up the 3D curves into smaller 3D curves so that the region to become a surfaced is obvious. The three curves do not need to be specified in any particular order.

The surface that is fit to the three bounding curves is a result of a transfinite interpolation, similar to the transfinite interpolation associated with the **tfi** command in the part phase.

This surface is similar to the **blend4** option.

Example

```
curd 1 csp3 00  -5.4e-01 3.6e-01
7.5e-01 6.1e-02  4.9e-01 8.6e-01
5.5e-01  3.2e-01 7.6e-01 ;;;
curd  2  csp3  00     -4.3e-01
4.8e-01    7.5e-01    -3.6e-01
-1.6e-02    9.2e-01    -4.1e-01
-5.9e-01    6.8e-01    -1.1e-01
-8.1e-01 5.7e-01 ;;;
curd 3 csp3 00 -2.7e-01 -7.9e-01
5.4e-01 2.3e-01 -6.2e-02 7.4e-01
4.0e-01  1.8e-01 8.9e-01 2.2e-02
7.8e-01 6.1e-01 ;;;
sd 1 blend3 2 1 3 ;
```

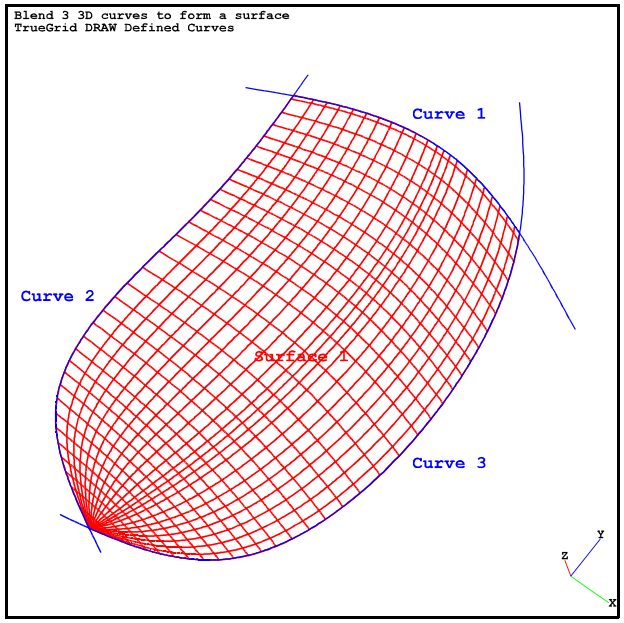


Figure 79 blended surface

blend4 blend four bounding 3D curves to form a patch

blend4 *3d_curve*$_1$ *3d_curve*$_2$ *3d_curve*$_3$ *3d_curve*$_4$ *trans* ;

where

3d_curve$_i$	3D curve number
trans	sequence of simple operators to transform the coordinates

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

The four 3D curves used to form the boundaries of a surface patch are assumed to intersect. Care must be taken to insure that the 3D curves almost intersect. The algorithm can deal with gaps between the 3D curves, but the result can be unsatisfactory if the gaps are very large. The curves do not need to end at the desired corner points of the surface patch.

It is also assumed that the four curves surround only one 4-sided region. If, due to curvature, there is an ambiguity as to which enclosed region will be surfaced, the results may be unsatisfactory. In such a case, we recommend that you break up the 3D curves into smaller 3D curves so that the region to be surfaced is obvious. The four curves do not need to be specified in any particular order.

The surface that is fit to the four bounding curves is a result of a transfinite interpolation, similar to the transfinite interpolation associated with the **tfi** command in the part phase.

Selecting 3D curves to form a four sided surface patch is an art. Ideally, opposite curves should be parallel. Avoid sharp bends in the curves. A kink in a boundary curve usually indicates the need for several surface patches using pieces of the original 3D curves. Severe concave boundary curves can also be a problem. If this occurs, create several more 3D curves to bisect the region and build several surface patches instead of one.

This surface is similar to the **blend3** option.

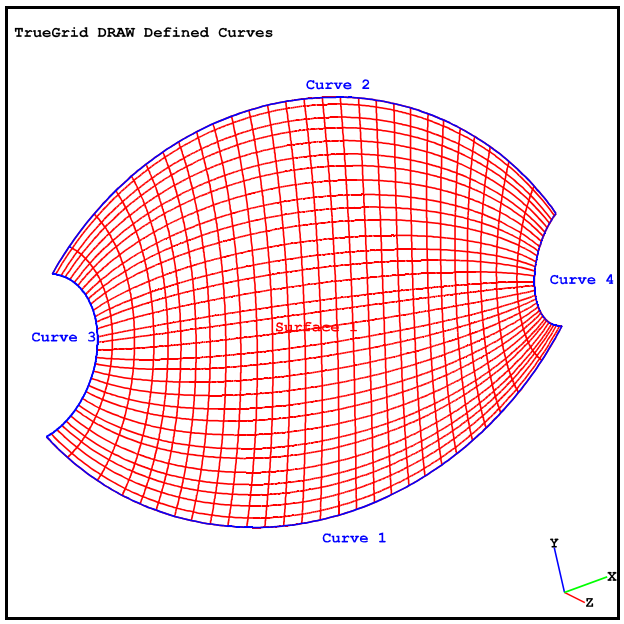


Figure 80 blended surface

Example

```
curd 1 arc3 seqnc   rt -6.7e+00 -5.2e-01  6.1e-01
                    rt  3.6e-01 -5.0e+00  3.1e-01
                    rt  6.4e+00 -2.5e+00  2.6e-01 ;
curd 2 arc3 seqnc   rt -5.7e+00  3.1e+00  8.2e-01
                    rt  4.0e-01  5.1e+00  5.9e-01
                    rt  6.6e+00  4.0e-01  7.5e-01 ;
curd 3 arc3 seqnc   rt -5.7e+00  3.1e+00  8.2e-01
                    rt -4.8e+00  1.3e+00  1.4e-01
                    rt -6.7e+00 -5.2e-01  6.1e-01 ;
curd 4 arc3 seqnc   rt  6.6e+00  4.0e-01  7.5e-01
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
                    rt  5.6e+00 -9.0e-01  9.7e-01
                    rt  6.4e+00 -2.5e+00  2.6e-01 ;
sd 1 blend4 1 2 3 4 ;
```

bsps B-spline

bsps *i-degree j-degree*

iknot$_1$ ... *iknot*$_{k1}$; *jknot*$_1$ .... *jknot*$_{k2}$;

xcontrol$_{1,1}$ *ycontrol*$_{1,1}$ *zcontrol*$_{1,1}$ ... *xcontrol*$_{1,n2}$ *ycontrol*$_{1,n2}$ *zcontrol*$_{1,n2}$

. . .

xcontrol$_{n1,1}$ *ycontrol*$_{n1,1}$ *zcontrol*$_{n1,1}$ ... *xcontrol*$_{n1,n2}$ *ycontrol*$_{n1,n2}$ *zcontrol*$_{n1,n2}$;

trans ;

where

i-degree j-degree	degrees of surface polynomial in the i and j direction.
i*knot*$_1$ ... *iknot*$_{k1}$	parametric coordinates of *k1* knots (0..) in the i direction.
j*knot*$_1$ ... *jknot*$_{k2}$	parametric coordinates of *k2* knots (0..) in the j direction.
xcontrol$_{1,1}$ *ycontrol*$_{1,1}$ *zcontrol*$_{1,1}$ ... *xcontrol*$_{1,n2}$ *ycontrol*$_{1,n2}$ *zcontrol*$_{1,n2}$ *xcontrol*$_{n1,1}$ *ycontrol*$_{n1,1}$ *zcontrol*$_{n1,1}$ ... *xcontrol*$_{n1,n2}$ *ycontrol*$_{n1,n2}$ *zcontrol*$_{n1,n2}$;	triplets of coordinates of control points

where

$n_1 = k_1$ - *multiplicity*$_1$

$n_2 = k_2$ - *multiplicity*$_2$

trans	sequence of simple operators to transform the coordinates

Remarks

This type of surface is also known as a Basis Spline because it is defined as a linear combination of polynomial basis functions. The theory of B-Splines can be found in many text books. The control points are the coefficients forming the linear combinations of the basis functions and can form a crude caricature of the surface.

The multiple knots have to be specified for the starting and ending point of the curve. The number of multiple knots is called multiplicity. The multiplicity for the starting and ending point determines the order of the curve. (2 - linear, 3 - quadratic, 4 - cubic ...).

multiplicity = degree + 1

The multiplicity has to be the same for the starting and ending points.

B-spline surfaces are also available through the IGES interface.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

The B-Spline surface, shown in **Figure 81**, is defined by the same control points as the NURBS Surface (**Figure 112**).

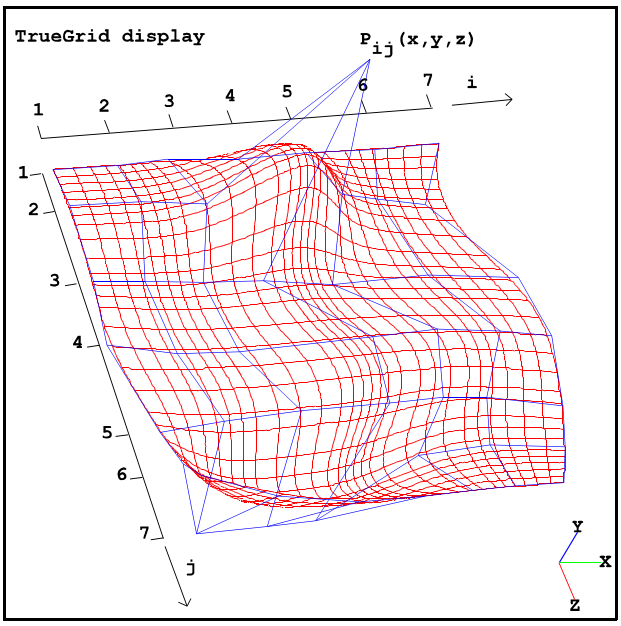


Figure 81 B-Spline Surface

```
sd 1 bsps 3 3
0 0 0 0 1 2 3 4 4 4 4;
0 0 0 0 1 2 3 4 4 4 4;
-2.17  4.90e-03 -1.80
-1.45  1.59e-02 -1.85
-8.54e-01      1.96e-02   -1.95
-4.35e-02  2.05e-02 -1.98
 8.01e-01  4.66e-03 -2.12
 1.65e+00 -5.72e-03 -2.19
 2.49e+00 -1.37e-03 -2.31
-2.14e+00  1.05e-02 -1.32
-1.35e+00  2.16e-02 -1.37
-6.19e-01      1.52e-02   -1.38
4.05e-01  2.00e+00 -1.50e+00
 1.01e+00  1.33e-04 -1.55e+00
                              1
                              .
                              41e+00  3.86e-03 -1.49e+00
 1.94e+00  2.21e-02 -1.40e+00
-2.17e+00 -2.57e-02 -3.81e-01 -1.58e+00 -6.88e-02 -4.22e-01
-9.27e-01 -1.13e-01 -4.49e-01 -2.38e-01 -1.24e-01 -4.82e-01
 5.19e-01 -9.57e-02 -4.24e-01  1.59e+00  1.64e-02 -4.53e-01
 2.60e+00  9.17e-02 -3.58e-01
-2.27e+00 -4.51e-02  3.97e-01 -1.48e+00 -1.52e-01 4.15e-01
-7.71e-01 -1.99e-01  3.57e-01  7.17e-01 -8.84e-02 2.68e-01
 1.43e+00  4.52e-03  2.52e-01  2.11e+00  6.75e-02 3.25e-01
 2.68e+00  9.55e-02  4.07e-01
-2.09e+00  3.17e-03  1.48e+00 -1.28e+00 -1.31e-01 1.26e+00
-3.79e-01 -1.57e-01  1.13e+00  5.72e-01 -6.70e-02 1.12e+00
 1.15e+00 -3.48e-03  1.11e+00  1.63e+00  3.45e-02 1.15e+00
 2.45e+00  5.85e-02  1.29e+00
-2.08e+00  1.66e-01  1.98e+00 -1.45e+00 -1.00e-00 1.93e+00
-6.71e-01 -1.00e-00  1.87e+00 -1.10e-01 -1.00e-00 1.83e+00
 5.99e-01 -2.52e-02  1.80e+00  1.42e+00  2.08e-02 1.73e+00
 2.28e+00  2.87e-02  1.78e+00
-2.03e+00  3.00e-01  2.21e+00 -1.14e+00  4.58e-02  2.29e+00
-2.59e-01 -2.78e-02  2.33e+00  4.70e-01 -1.12e-02 2.29e+00
 1.02e+00  4.84e-03  2.26e+00  1.52e+00  7.42e-03 2.221e+00
 2.10e+00  4.38e-04  2.19e+00 ;;
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

bstl read the standard binary STL file

bstl *file trans* ;

where

file	path and file name (case sensitive in UNIX/LINUX)
trans	sequence of simple operators to transform the coordinates

Remarks

This command permits you to import triangular surfaces via StereoLithography (STL) files.

Use the **fetol** command before this command to extract interior features as edges.

Example

```
fetol 110
sd 1 bstl  prt_k1.bstl ;
```

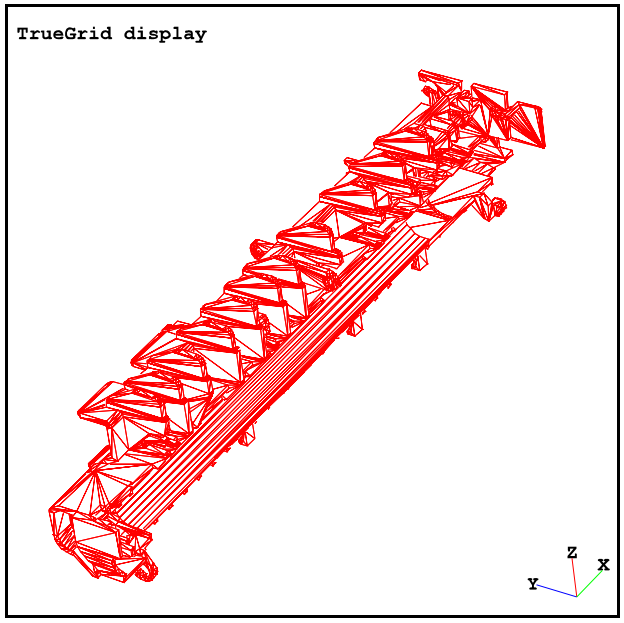


Figure 82 Surface imported from binary stl

cn2p infinite cone, defined by two points

cn2p $x_0\ y_0\ z_0\ x_n\ y_n\ z_n\ r_1\ t_1\ r_2\ t_2$

where

x_0	first coordinate of a point on the axis of rotation
y_0	second coordinate of a point on the axis of rotation
z_0	third coordinate of a point on the axis of rotation
x_n	first component of axis direction vector
y_n	second component of axis direction vector
z_n	third component of axis direction vector
r_1	radius at the cross section along axis at t_1
t_1	position along axis with radius r_1
r_2	radius at the cross section along axis at t_2
t_2	position along axis with radius r_2

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

(x_0,y_0,z_0) and (x_n,y_n,z_n) define the axis of symmetry. You complete the cone's definition by specifying two points (r_1,t_1) and (r_2,t_2) in any planar cross section containing the axis of symmetry. The t coordinate is measured along the axis from the point (x_0,y_0,z_0) in the direction of (x_n,y_n,z_n). The r coordinate is measured orthogonally from the axis of symmetry. For any plane containing the axis of symmetry, the line segment formed by the two points in the plane will be contained on the cone. The coordinate pairs (r_1,t_1) and (r_2,t_2) must satisfy the following conditions:

$$|r_1 - r_2| \times 10^{-18} < |t_1 - t_2|$$

and

$$|t_1 - t_2| \times 10^{-18} < |r_1 - r_2|$$

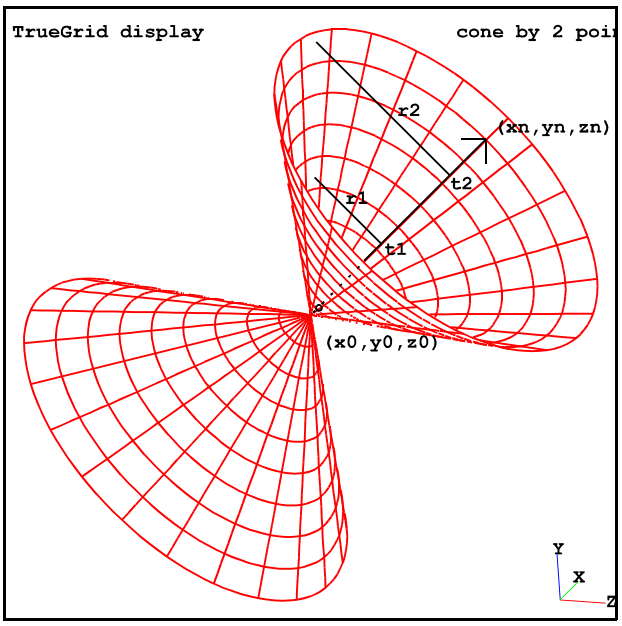


Figure 83 Cone by an axis and two points

A point projected to this surface should not start out on the axis of symmetry. See the example in **Figure 83**. This is an infinite surface. The graphics will only show a portion of the surface. That portion shown in the graphics changes as the objects in the picture change.

Example

```
sd 2 cn2p 0 0 0 -1 1 1 1 1 2 2
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

cone infinite cone, defined by radius and angle

cone $x_0\ y_0\ z_0\ x_n\ y_n\ z_n\ r\ \theta$

where

x_0	first coordinate of a point on the axis of rotation
y_0	second coordinate of a point on the axis of rotation
z_0	third coordinate of a point on the axis of rotation
x_n	first component of axis direction vector
y_n	second component of axis direction vector
z_n	third component of axis direction vector
r	radius of the cross section along axis at the point $(x_0\ y_0\ z_0)$
θ	angle of the cone

Remarks

This command defines a cone by specifying its radius and angle. The two points (x_0,y_0,z_0) and (x_n,y_n,z_n) define the axis of symmetry. The plane orthogonal to the axis at the point (x_0,y_0,z_0), slices through the cone to form a circle of radius r. r must be non-negative. Note that if $r = 0$, then the point (x_0,y_0,z_0) is at the apex of the cone. The cone forms an angle θ relative to its axis of symmetry. This angle must be between -90 and 90, excluding -90, 0, and 90. A point projected to this surface can not be on the axis of symmetry. This is an infinite surface. The graphics will only show a portion of the surface. The portion shown in the graphics changes as the objects in the picture change.

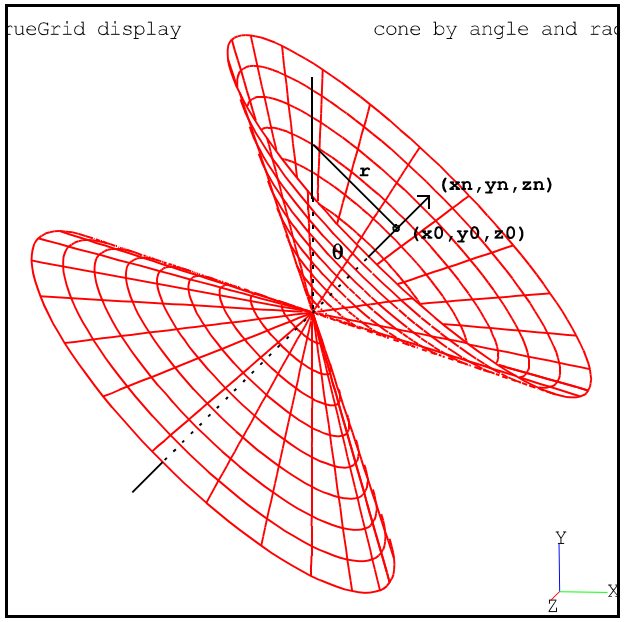


Figure 84 Cone specified by a radius and an angle

Example

```
sd 7 cone 0 1 2 1 1 0 1 45
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

cp infinite generalized cylinder (extruded or lofted curve)

cp *ln trans* ;

where

ln	2D curve number
trans	sequence of simple operators to transform the coordinates

Remarks

This option enables you to define a surface by sweeping a planar 2D curve through the third dimension, then transforming the result. The curve *ln* lies in the xz-plane and then is extended infinitely in the y direction to form a generalized cylinder.

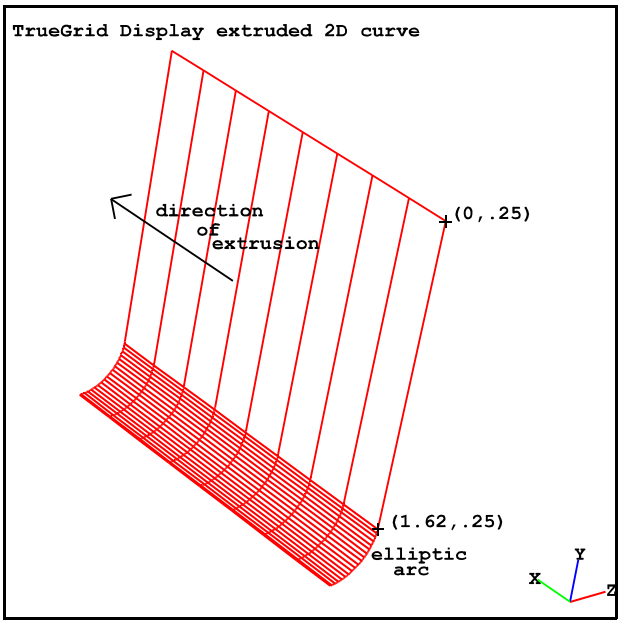


Figure 85 2D curve extruded in the x-direction

Example

```
ld 4 lp2 0 .25 1.62 .25;lep .25
.25 1.62 0 90 0 0 ;
sd 3 cp 4 rz -90;
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

cr 2D curve revolved about an axis

cr $x_0\ y_0\ z_0\ x_n\ y_n\ z_n\ ln$

where

x_0	first coordinate of a point on the axis of rotation
y_0	second coordinate of a point on the axis of rotation
z_0	third coordinate of a point on the axis of rotation
x_n	first component of axis direction vector
y_n	second component of axis direction vector
z_n	third component of axis direction vector
ln	2D curve number

Remarks

This command forms a surface by rotating the 2D curve *ln* about an axis given by (x_0,y_0,z_0) and (x_n,y_n,z_n). As an example, the following commands were used to generate the picture below. The 2D curve must be defined before it is referenced by this command. The numbers in the picture correspond to the points in the 2D curve definition. A point projected to this surface should be away from the axis of symmetry.

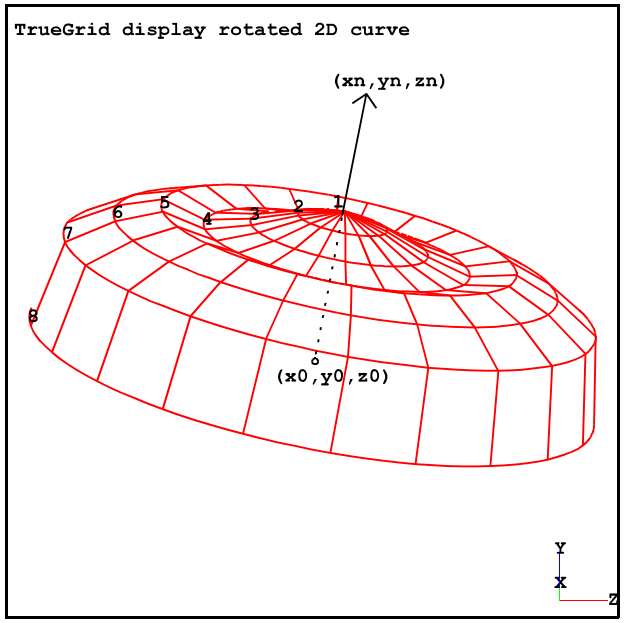


Figure 86 2D curve rotated about an axis

Example

```
ld 1 lp2 0 .2266 .0625 .2109
.125 .1875 .1875 .1719 .25 .1875
.3125 .164 .375 .125 .4 0;
sd 1 cr 0 4 1 0 1 .2 1
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

crule3d cylindrical surface between two 3D curves

crule3d *3D-curve*$_1$ *3D-curve*$_2$ *trans ;*

where

3D-curve$_1$	first boundary 3D curve
3D-curve$_2$	second boundary 3D curve
trans	sequence of simple operators to transform the coordinates

Remarks

This command forms a surface using linear interpolation in cylindrical coordinates between two 3D curves. At least one of the two curves must have a positive arc length. Both curves must be already defined. 3D curves are defined with the **curd** command or read from an **IGES** file. The direction of the curves is critical to the interpolation of the surface. Check the order of the points in each of the curves by displaying the points with the **label** command. In order to reverse the order of a curve, specify the curve definition number with a negative number. The surface is constructed by pairing each point on one curve with a point on the other curve. These two points are connected by a line segment in cylindrical coordinates. Points along the two curves are paired by relative arc length.

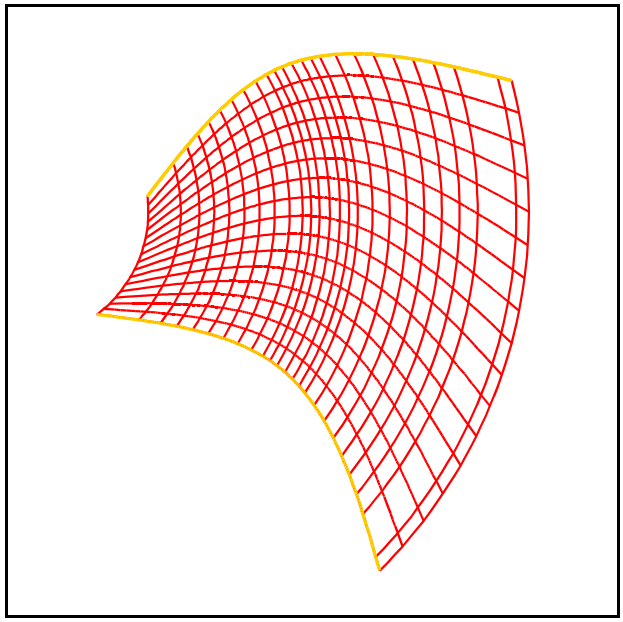


Figure 87 Cylindrical ruled surface

Example

```
curd 1 csp3 00 .3 -.4 0 1 -.6 0 1.4 -1.4 0;;
curd 2 cpcd 1 rz 60;
sd 1 crule3d 1 2 ; ;
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

crx rotate 2D curve about x-axis

crx *ln*

where *ln* is the 2D curve number

Remarks

The 2D curve *ln* must be defined before it is referenced by this command using the **ld** command. A point projected to this surface should be away from the axis of symmetry. The following commands were used to generate the surface in the picture below. The first two points of the 2D curve are labeled in the picture. A circular arc is appended to the line segment between the two points.

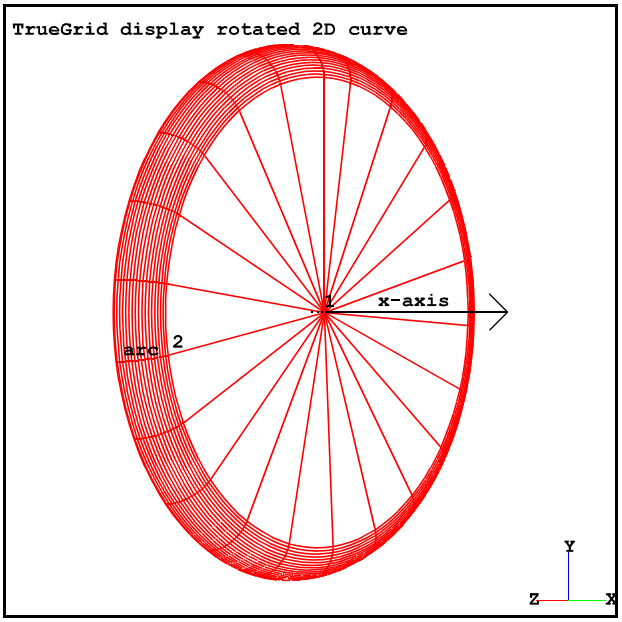


Figure 88 2D curve rotated about the x-axis

Example

```
ld 4 lp2 0 .25 1.62 .25; lep .25
.25 1.62 0 90 0 0 ;
sd 3 crx 4
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

cry rotate 2D curve about y-axis

cry *ln*

where *ln* is the 2D curve number

Remarks

This command defines a surface by rotating a 2D curve about the y-axis. The 2D curve *ln* must be defined before it is referenced by this command using the **ld** command. A point projected to this surface should be away from the axis of symmetry. The following commands were used to create the surface in the picture below. The four points in the **lp2** command are annotated in the picture (**Figure 89**).

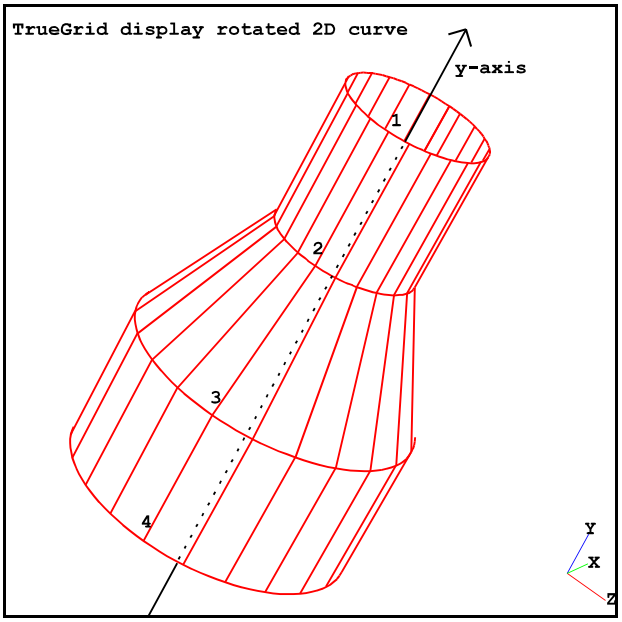


Figure 89 2D curve rotated about the y-axis

Example

```
ld 1 lp2 1 1 1 2 .5 3 .5 4;
sd 1 cry 1
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

crz rotate 2D curve about z-axis

crz *ln*

where *ln* is the 2D curve number

Remarks

This command defines a surface by rotating a 2D curve about the z-axis. The 2D curve *ln* must be defined before it is referenced by this command using the **ld** command. A point projected to this surface should be away from the axis of symmetry. The following commands created the surface in the picture below. The two points in the **lp2** command are annotated.

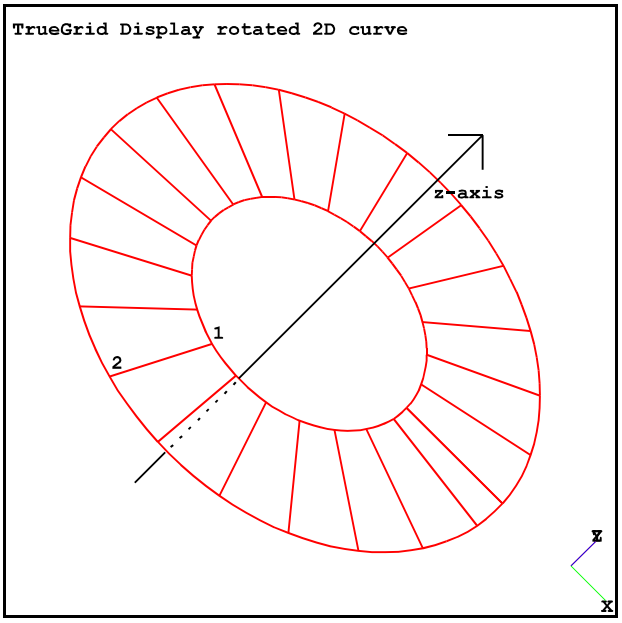


Figure 90 2D curve rotated about the z-axis

Example

```
ld 1 lp2 1 1 2 1;
sd 1 crz 1
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

csps fit a cubic spline surface through 3D data

csps *#_columns #_rows flag conditions* $x_{11}\, y_{11}\, z_{11}\, x_{12}\, y_{12}\, z_{12}$ ... *trans ;*

where

#_columns	number of control points in each column of the table of control points
#_rows	number of control points in each row of the table of control points
flag	4 digits formed using **0** for natural boundary condition, **1** for specified boundary derivatives, one digit for each of the four edges
conditions	specifies the derivatives along the edges, in order, when a flag of 1 is used in the corresponding digit in the *flag*
where each *condition* is formed by derivatives	
dx_{ij}, dy_{ij}, dz_{ij}	derivative triplet, one for each control point along the edge
$x_{ij}\, y_{ij}\, z_{ij}$	list of coordinates for the control points on the surface
trans	sequence of simple operators to transform the coordinates

Remarks

There are four binary digits forming the flag. Each digit corresponds to an edge of the surface. The first edge is formed by the first column. The second edge is formed by the last column. Edge 3 is the first row. Edge 4 is the last row.

The theory of cubic splines allows for a derivative to be specified at each end of a spline curve. In a cubic spline surface, derivatives can be specified at both ends of each row and column. Each derivative is specified as a vector with three components, *dx*, *dy*, and *dz*. The magnitude of the derivatives can have a significant affect on the shape of the surface. Experimentation may be required. If derivatives are not specified, then the natural derivative is used. This means that the second derivative is set to zero which uses the remaining degrees of freedom with no first derivatives needed.

In a laborious way, one can form a cubic spline surface from a set of 3D cubic spline curves. Using a text editor on the tsave file that recorded the definition of the 3D curves, one can cut and paste to form the table of control points for the cubic spline surface. This requires that all of the 3D curves have the same number of control points and oriented in the same general direction. The curve control points have to placed into the table in the proper order. Derivatives at the end control points (if any) become the derivatives along an edge. The ordering is critical. Care is also needed in selecting corresponding interior control points in each 3D curve that represent corresponding features in the shape of the surface.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

The transformation *trans* can include the normal offset. The normal offset must be the first operator in the list of operators that form the transformation. This normal offset is intended for small offsets only, otherwise the resulting surface will be self-intersecting.

Examples

In this first example, one of the edges is degenerate and all of the edge derivatives are specified. In this case, the end derivatives of the degenerate edge are zero.

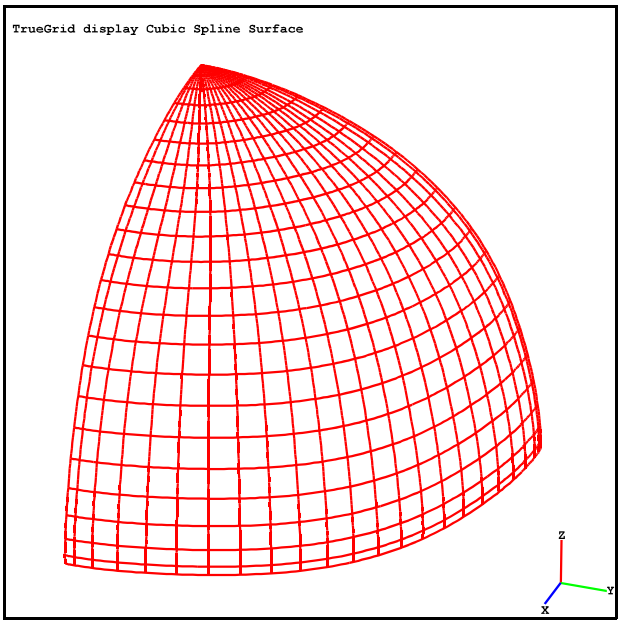


Figure 91 Spline surface with degenerate edge

```
c  The magnitude of the derivatives are parametric
para am1 1 am2 1 am3 1 am4 1.3;
c  All derivatives are specified in this example
sd 1 csps 4 4 1111
c  x,y,z components of the derivatives on edge 1
0 %am1 0 0 %am1 0 0 %am1 0 0 0 0
c  x,y,z components of the derivatives on edge 2
[-%am2] 0 0 [-%am2] 0 0 [-%am2] 0 0 0 0 0
c  x,y,z components of the derivatives on edge 3
0 0 %am3 0 0 %am3 0 0 %am3 0 0 %am3
c  x,y,z components of the derivatives on edge 4
[-%am4] 0 0 [-%am4*cos(30)] [-%am4*sin(30)] 0
[-%am4*cos(60)] [-%am4*sin(60)] 0 0 [-%am4] 0
c  x,y,z control points for row 1
1 0 0 [cos(30)] [sin(30)] 0 [cos(60)] [sin(60)] 0 0 1 0
c  x,y,z control points for row 2
[sqrt(3)/2] 0 .5 [cos(30)*sqrt(3)/2] [sin(30)*sqrt(3)/2] .5
[cos(60)*sqrt(3)/2] [sin(60)*sqrt(3)/2] .5 0 [sqrt(3)/2] .5
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
c  x,y,z control points for row 3
.5 0 [sqrt(3)/2] [cos(30)/2] [sin(30)/2] [sqrt(3)/2]
[cos(60)/2] [sin(60)/2] [sqrt(3)/2] 0 .5 [sqrt(3)/2]
c  x,y,z control points for row 4
0 0 1 0 0 1 0 0 1 0 0 1 ;;
```

```
sd 1 csps 5 5 0000
c number of control points
c in the i and j-direction
c(rows and columns)
c Natural Boundary Conditions
1 1 1 2 1 1 2.5 1 1 3 1 1 4 1 1
1 2 1 1.79 1.79 2 2.5 1.5 2
3.2 1.79 2 4 2 1 1 2.5 1
1.5 2.5 2 2.5 2.5 1 3.5 2.5 2
4 2.5 1 1 3 1 1.79 3.2 2
2.5 3.5 2 3.2 3.2 2 4 3 1 1 4 1
2 4 1 2.5 4 1 3 4 1 4 4 1;;
```

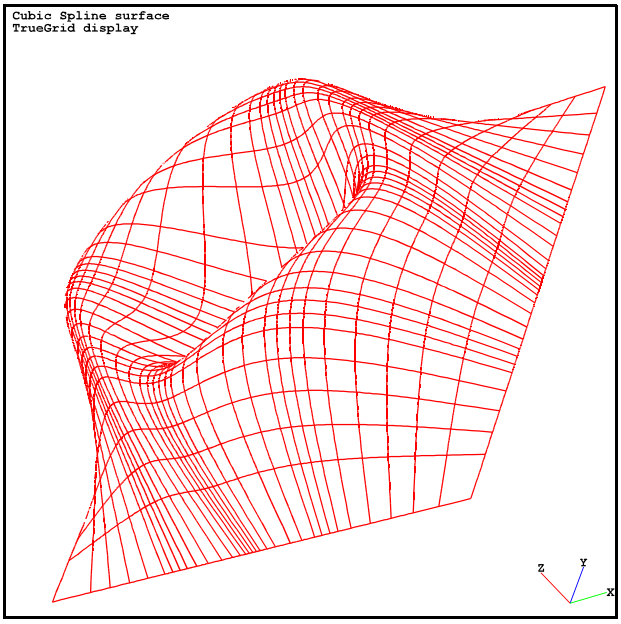


Figure 92 cubic spline surface

The last example is used with a normal of 1 applied to it to form the next surface. Care is needed in using the normal offset. The normal distance must not exceed the inverse of the curvature or the resulting surface will be self-intersecting.

```
sd 1 csps 5 5 0000
1 1 1 2 1 1 2.5 1 1 3 1 1 4 1 1
1 2 1 1.79 1.79 2 2.5 1.5 2
3.2 1.79 2 4 2 1 1 2.5 1
1.5 2.5 2 2.5 2.5 1 3.5 2.5 2
4 2.5 1 1 3 1 1.79 3.2 2
2.5 3.5 2 3.2 3.2 2 4 3 1 1 4 1
2 4 1 2.5 4 1 3 4 1 4 4 1 normal
1;;
```

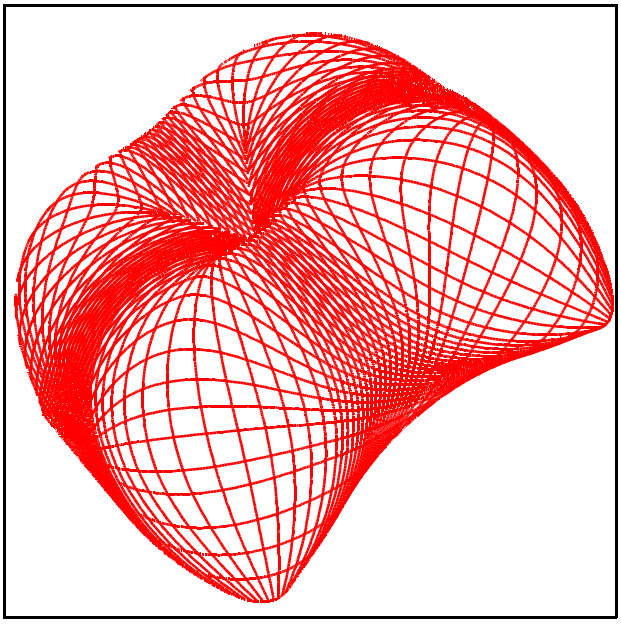


Figure 93 Cubic spline surface w/ derivatives

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

In this next example, two surfaces are constructed from 3D cubic spline curves. All of the control points are from the unit sphere. The boundary control points are the same for both surfaces. Only the interior control point differs. This pair of surfaces is intended to demonstrate the need to select points in the cross section curves so that they correspond to the same features. But this example also demonstrates the absurdity of trying to approximate a sphere with a cubic spline surface with only 9 control points. It also shows that the boundary edges do not follow the 3D cubic spline curves that pass through the same control points.

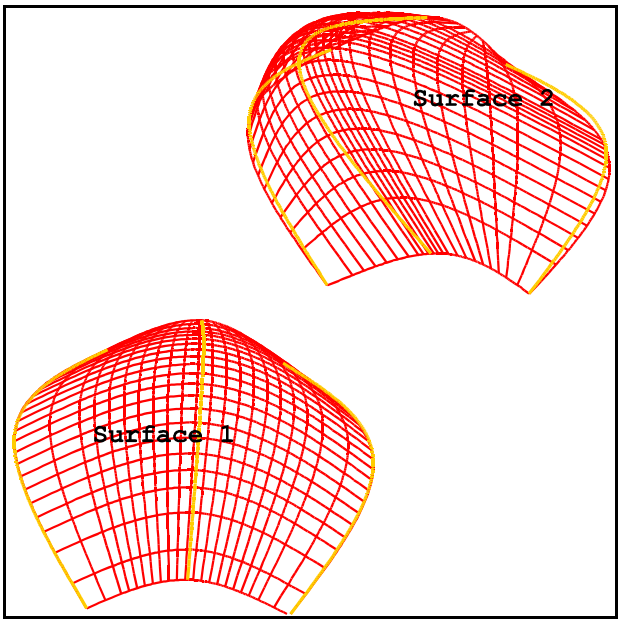


Figure 94 Cubic spline surface from curves

```
curd 1 csp3 00
  -5.2497667e-01  -2.9622331e-07
8.5111660e-01
  -1.0000000e+00  -5.6425995e-07
2.2724271e-07
 -5.2497709e-01 -2.9622353e-07 -8.5111636e-01 ;;;
curd 2 csp3 00
 -2.6248786e-01 -4.5464340e-01 8.5111660e-01
 -4.9999908e-01 -8.6602592e-01 2.2724271e-07
 -2.6248807e-01 -4.5464376e-01 -8.5111636e-01 ;;;
curd 3 csp3 00
 2.4999917e-01 -4.3301326e-01 8.6602539e-01
 4.9999827e-01 -8.6602640e-01 2.2724271e-07
 2.8678745e-01 -4.9673271e-01 -8.1915170e-01 ;;;
sd 1 csps 3 3 0000
 -5.2497667e-01 -2.9622331e-07 8.5111660e-01
 -1.0000000e+00 -5.6425995e-07 2.2724271e-07
 -5.2497709e-01 -2.9622353e-07 -8.5111636e-01
 -2.6248786e-01 -4.5464340e-01 8.5111660e-01
 -4.9999908e-01 -8.6602592e-01 2.2724271e-07
 -2.6248807e-01 -4.5464376e-01 -8.5111636e-01
 2.4999917e-01 -4.3301326e-01 8.6602539e-01
 4.9999827e-01 -8.6602640e-01 2.2724271e-07
 2.8678745e-01 -4.9673271e-01 -8.1915170e-01 ;;;
curd 4 csp3 00
 -5.2497667e-01 -2.9622331e-07 8.5111660e-01
 -1.0000000e+00 -5.6425995e-07 2.2724271e-07
 -5.2497709e-01 -2.9622353e-07 -8.5111636e-01 ;;;
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
curd 5 csp3 00
 -2.6248786e-01 -4.5464340e-01 8.5111660e-01
 -7.5146174e-01 -2.2497362e-01 6.2023556e-01
 -2.6248807e-01 -4.5464376e-01 -8.5111636e-01 ;;;
curd 6 csp3 00
 2.4999917e-01 -4.3301326e-01 8.6602539e-01
 4.9999827e-01 -8.6602640e-01 2.2724271e-07
 2.8678745e-01 -4.9673271e-01 -8.1915170e-01 ;;;
sd 2 csps 3 3 0000
 -5.2497667e-01 -2.9622331e-07 8.5111660e-01
 -1.0000000e+00 -5.6425995e-07 2.2724271e-07
 -5.2497709e-01 -2.9622353e-07 -8.5111636e-01
 -2.6248786e-01 -4.5464340e-01 8.5111660e-01
 -7.5146174e-01 -2.2497362e-01 6.2023556e-01
 -2.6248807e-01 -4.5464376e-01 -8.5111636e-01
 2.4999917e-01 -4.3301326e-01 8.6602539e-01
 4.9999827e-01 -8.6602640e-01 2.2724271e-07
 2.8678745e-01 -4.9673271e-01 -8.1915170e-01 ;;;
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

cy infinite cylinder

cy $x_0\ y_0\ z_0\ x_n\ y_n\ z_n$ *radius*

where

$x_0\ y_0\ z_0$	point on the cylinder's axis
$x_n\ y_n\ z_n$	direction vector which defines the cylinder's axis
radius	radius of the cylinder

Remarks

This command defines a cylinder with radius, *radius*, and an axis given by (x_0,y_0,z_0) and (x_n,y_n,z_n), as shown in the picture below. Specifically, (x_0,y_0,z_0) is any point on the cylinder's axis and the cylinder's axis is parallel to the vector which extends from the origin to the point (x_n,y_n,z_n). The radius of the cylinder must be positive. When projecting a point onto this surface, be sure that it is initialized somewhere away from the axis of symmetry to make it clear which direction to project. This is an infinite surface. The graphics will only show a portion of the surface. That portion shown in the graphics changes as the objects in the picture change.

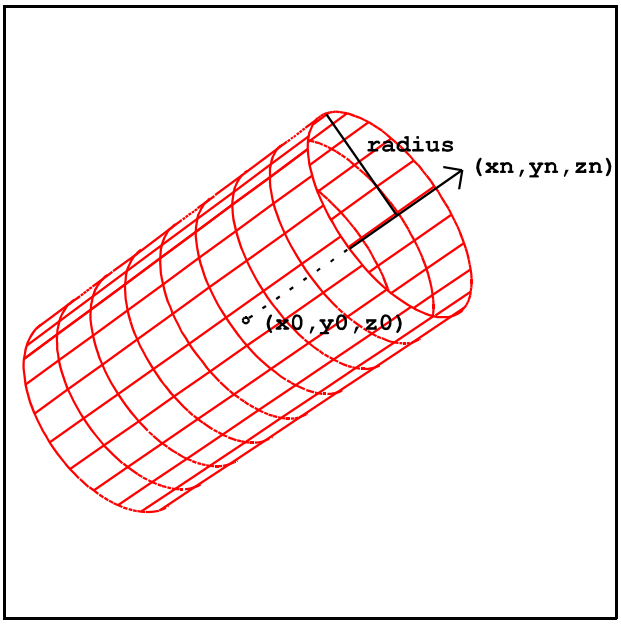


Figure 95 Cylinder by an axis and a radius

Example

```
sd 5 cy 0 0 0 1 0 0 12.5
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

cy3 cylinder from 3 points

cy3 $x_1\ y_1\ z_1\ x_2\ y_2\ z_2\ x_3\ y_3\ z_3$

where

$(x_1\ y_1\ z_1)$	global coordinates of the first point
$(x_2\ y_2\ z_2)$	global coordinates of the second point
$(x_3\ y_3\ z_3)$	global coordinates of the third point

Remarks

The three points are assumed to form a cross section of the cylinder, perpendicular to the axis of symmetry.

Example

```
curd 1 arc3 whole rt .1 .2 .3
                  rt -.2 .3 1
                  rt -1 -2 -1 ;
sd 1 cy3
 -.9190374 -.05553633 1.7810655
    -.23211999    -1.1207378
-1.0815152
 -2.073 -2.403686 .35669363
```

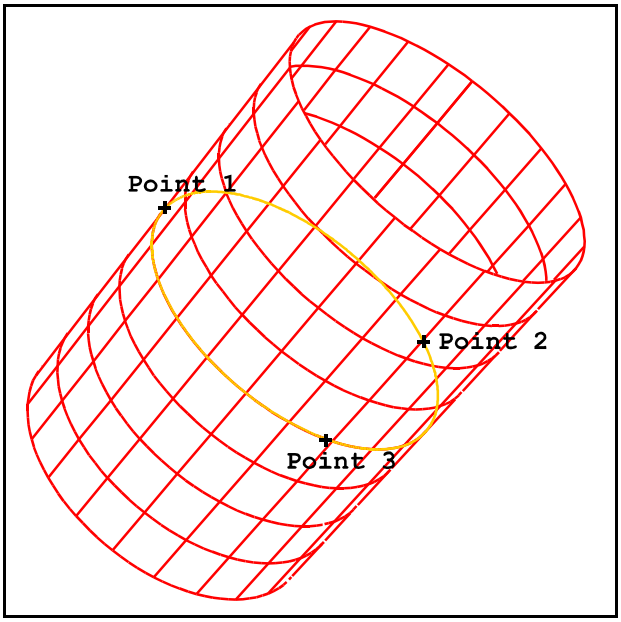


Figure 96 Cylinder by 3 points

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

cyr2 cylinder from 2 points and radius

cyr2 $x_1\ y_1\ z_1\ x_2\ y_2\ z_2\ radius$

where

$(x_1\ y_1\ z_1)$	global coordinates of the first point
$(x_2\ y_2\ z_2)$	global coordinates of the second point
radius	radius of the cylinder

Remarks

The two points are used to establish the axis of rotation. They are selected in global Cartesian coordinates.

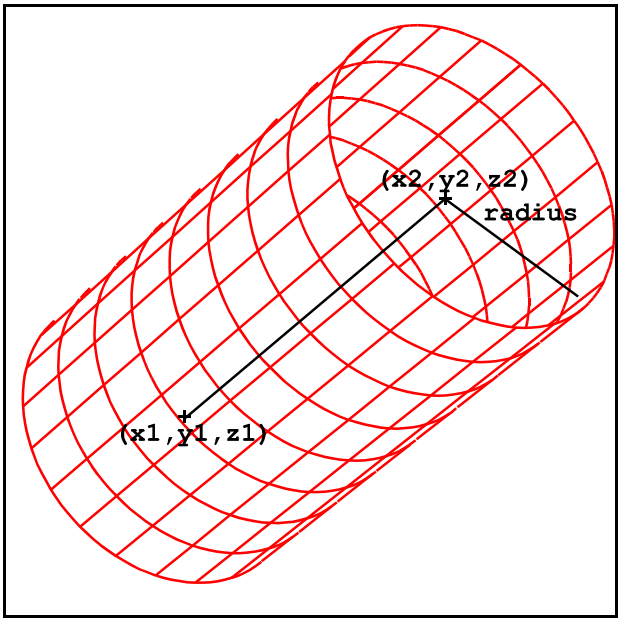


Figure 97 2 points form the axis

Example

```
sd 1 cyr2 1 2 3 1.5 4 5 1 ;
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

cyr3 cylinder from 3 points and radius

cyr3 $x_1\ y_1\ z_1\ x_2\ y_2\ z_2\ x_3\ y_3\ z_3$ *radius*

where

$(x_1\ y_1\ z_1)$	global coordinates of the first point
$(x_2\ y_2\ z_2)$	global coordinates of the second point
$(x_3\ y_3\ z_3)$	global coordinates of the third point
radius	radius of the cylinder

Remarks

Three points are selected on a planar cross section, perpendicular to the axis, are used to establish the axis. The radius of the cylinder that passes through these three points is ignored and uses the one specified. This makes it easy to create a plug to a hole with a gap between the plug and the hole. The points are selected in global Cartesian coordinates.

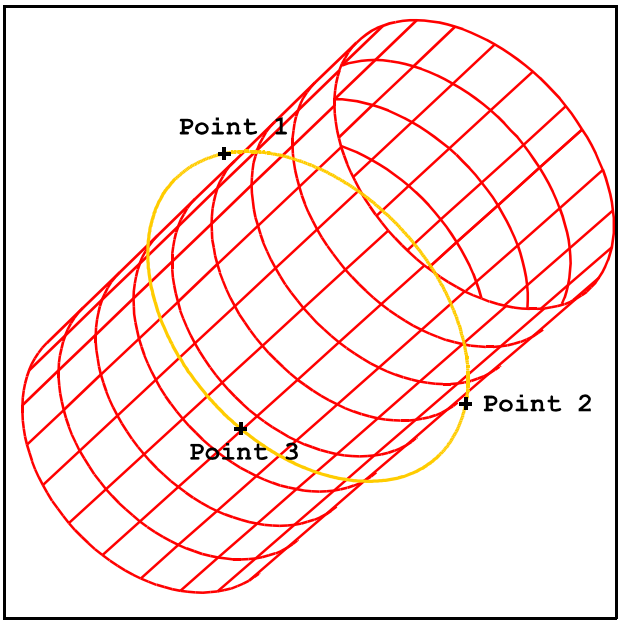


Figure 98 3 points from a circle

Example

```
curd 1 arc3 whole rt .1 .2 .3
                  rt -.2 .3 1
                  rt -1 -2 -1 ;
circent
 -.9190374 -.05553633 1.7810655
    -.23211999    -1.1207378
-1.0815152
 -2.073 -2.403686 .35669363
c  center =  -1.01674E+00  -1.07451E+00   4.24898E-01
c  radius =    1.6991307
c  normal =  -7.11559E-01   5.85396E-01  -3.88582E-01
sd 1 cyr3
 -.9190374 -.05553633 1.7810655
 -.23211999 -1.1207378 -1.0815152
 -2.073 -2.403686 .35669363
 [1.6991307-.25]
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

er ellipsoid (ellipse revolved about an axis)

er $x_0\ y_0\ z_0\ x_n\ y_n\ z_n\ r_1\ r_2$

where

x_0	first coordinate of a point on the axis of rotation
y_0	second coordinate of a point on the axis of rotation
z_0	third coordinate of a point on the axis of rotation
x_n	first component of axis direction vector
y_n	second component of axis direction vector
z_n	third component of axis direction vector
r_1	radius in the axial direction
r_2	radius orthogonal to the axis

Remarks

An ellipse is rotated about an axis given by (x_0,y_0,z_0) and (x_n,y_n,z_n). To specify the shape of the ellipse, you give the lengths of its two axes: r_1 is the length of the axis perpendicular to the axis of rotation, and r_2 is the length of the axis along the axis of rotation. Both lengths must be positive. A point projected onto this surface from within should be away from the axis of symmetry. See **Figure 99**.

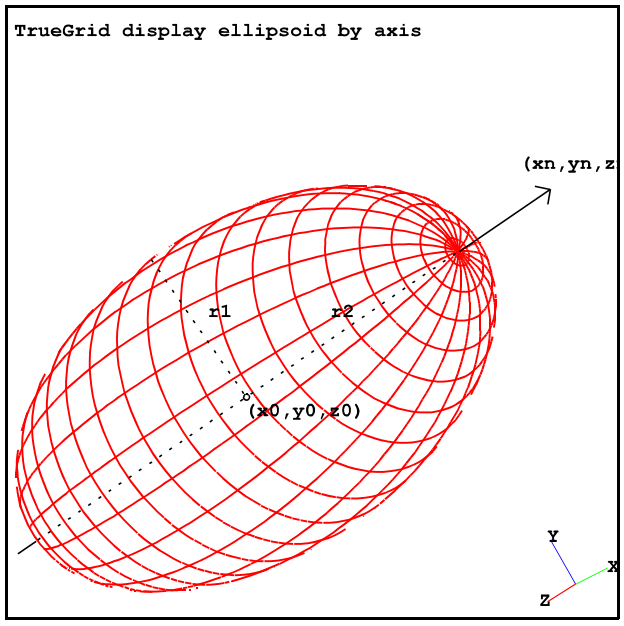


Figure 99 ellipsoid by axis and two dimensions

Example

```
sd 6 er 0 0 0 1 0 0 1 2
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

face face of the present part (Part Phase only)

face *region*

Remarks

The region should be a face of the current part; the surface is defined to match the face as shown in **Figure 100**. This surface becomes available after the **endpart** command finishes the definition of the part.

Use the **fetol** command before this command to extract interior features as edges.

Example

```
sd 1 face 1 2 1 1 5 6
```

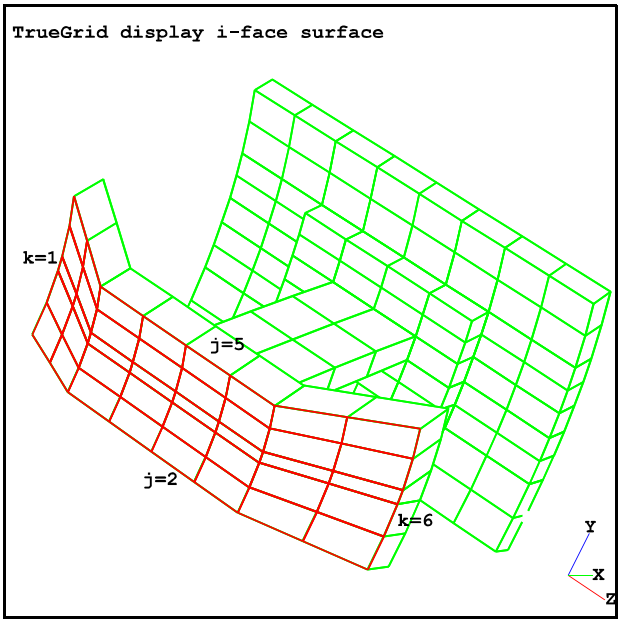


Figure 100 Face of a Part as a Surface

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

faceset convert a face set into a surface

faceset *set_name*

where *set_name* is the name of a face set

Remarks

This option is only available in the merge phase. A face set can be created using the set operations in the part or merge phase. The part phase method is parametric and the merge phase is very general. The command to use is **fset**. Alternatively, select the ***Pick*** button in the merge phase environment window followed by the ***Sets*** and ***Faces*** buttons. Use the lasso to select the faces. These methods can be combine to form the desired face set. The set is then easily converted to a surface with this command.

When 8-noded faces are used, they are converted to four 4-noded faces using the centroid as a 9th node.

Use one of the merging commands such as **stp** before using this command. This causes near duplicate nodes to be removed and the surface will have no extra or strange interior edges.

Use the **fetol** command before this command to extract interior features as edges.

This surface is ideal when a hex mesh is to be attached to an existing model. For example, a NASTRAN model can be read in and the exterior of that mesh can be converted to a surface using this method. This is required when using the **blude** part command.

Example

A NASTRAN mesh of a simple ship is import using the **readmesh** command and one of the sides of the ship (wet surface only) is selected using the interactive set creation features. This set is then converted to a surface using:

```
sd faceset starboard

c starboard is the face set highlighted in red.
c These faces where selected using the
c lasso after selecting pick, sets, and faces.
```

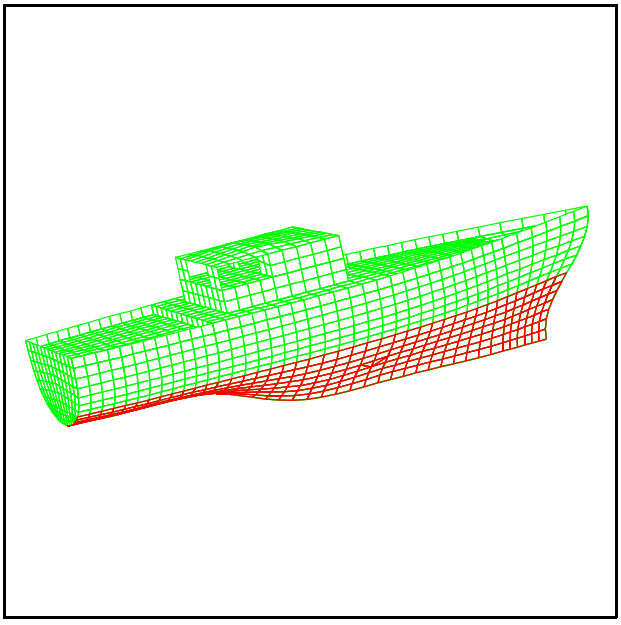


Figure 101 Faceset surface

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

function surface by three algebraic expressions

function u_{min} u_{max} v_{min} v_{max} *x-expression ; y-expression ; z-expression ; trans ;*

where

u_{min}	minimum value of the first independent variable
u_{max}	maximum value of the first independent variable
v_{min}	minimum value of the second independent variable
v_{max}	maximum value of the second independent variable
x-expression	algebraic expression for first 3D coordinate
y-expression	algebraic expression for second 3D coordinate
z-expression	algebraic expression for third 3D coordinate
trans	sequence of simple operators to transform the coordinates

Remarks

Function defines a surface with algebraic forms referencing independent variables u and v in the rectangle:

$$u_{min} < \mathbf{u} < u_{max}, \qquad v_{min} < \mathbf{v} < v_{max}.$$

Three Fortran-like expressions map the rectangle to the coordinates x, y, and z in the global coordinate system.

Example

```
sd 2 function -1 1 -1 1
u;v;u*u-v*v;;
```

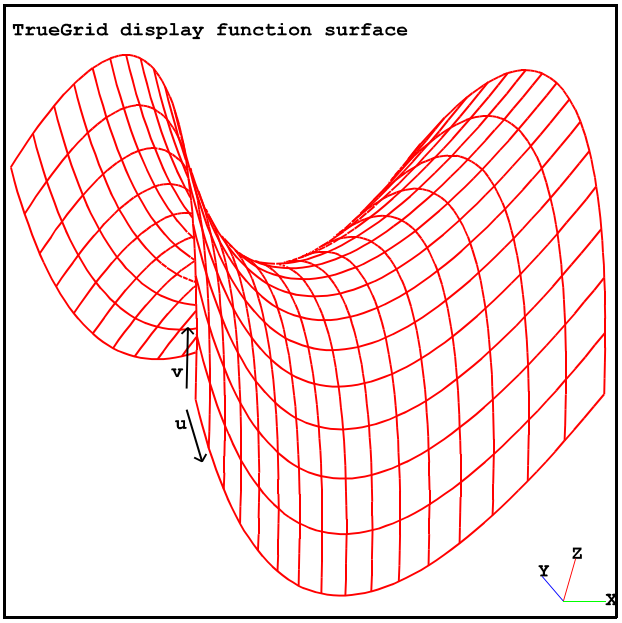


Figure 102 Saddle surface using functions

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

hermite precision 2nd order spline surface

hermite #_*columns* #_*rows flag conditions* $x_{11}\, y_{11}\, z_{11}\, x_{12}\, y_{12}\, z_{12}$ ... *trans* ;

where

#_columns	number of control points in each column of the table of control points
#_rows	number of control points in each row of the table of control points
flag	4 digits formed using **0** for natural boundary condition, **1** for specified boundary derivatives, **2** for a loop, one digit for each of the four edges
conditions	specifies the derivatives along the edges, in order, when a flag of 1 is used in the corresponding digit in the *flag*
where each *condition* is formed by derivatives	
dx_{ij}, dy_{ij}, dz_{ij}	derivative triplet, one for each control point along the edge
$x_{ij}\, y_{ij}\, z_{ij}$	list of coordinates for the control points on the surface
trans	sequence of simple operators to transform the coordinates

Remarks

This is designed to place a smooth surface through a very large number of regularly spaced points. It has 2 derivatives at each point. The resulting surface is very accurate in that it passes approximately through all of the points within a very small tolerance. This surface is different from a cubic spline surface in that it is not 2 times differentiable everywhere. This 2nd derivative, which is typically found in a cubic spline surface, has been sacrificed to get a surface that is very fast and very accurate. It can easily handle millions of control points and has at least 1 derivative everywhere. The form of this data for this surface is identical to a cubic spline surface, so, if after trying a cubic spline and not getting the desired results, you can easily try this type of spline surface.

A normal offset can be applied to this surface by issuing the normal option followed by the offset as the first operator in the transformation *trans*.

Use the **wiges** command to write surfaces of **hermite** type to an IGES file as a parametric spline surface.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Examples

In the examples below, all have the same set of control points. Only the edge derivatives are different.

In this first example, the natural derivatives are used.

```
sd 1 hermite 4 3 0000
 2 0 0 0 2 0 -2 0 0 0 -2 0
 1 0 1 0 1 1 -1 0 1 0 -1 1
 2 0 2 0 2 2 -2 0 2 0 -2 2 ; ;
```

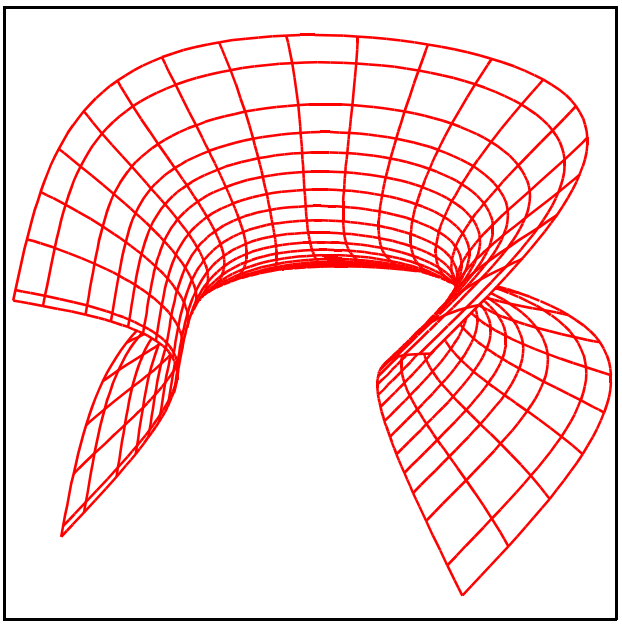


Figure 103 Natural derivatives

In the next example, derivatives have been selected. The magnitude of the derivatives can be changed to vary the effect of the derivatives.

```
para d 10;
sd 1 hermite 4 3 1100
 0 %d 0 0 %d 0 0 %d 0
 %d 0 0 %d 0 0 %d 0 0
 2 0 0 0 2 0 -2 0 0 0 -2 0
 1 0 1 0 1 1 -1 0 1 0 -1 1
 2 0 2 0 2 2 -2 0 2 0 -2 2 ; ;
```

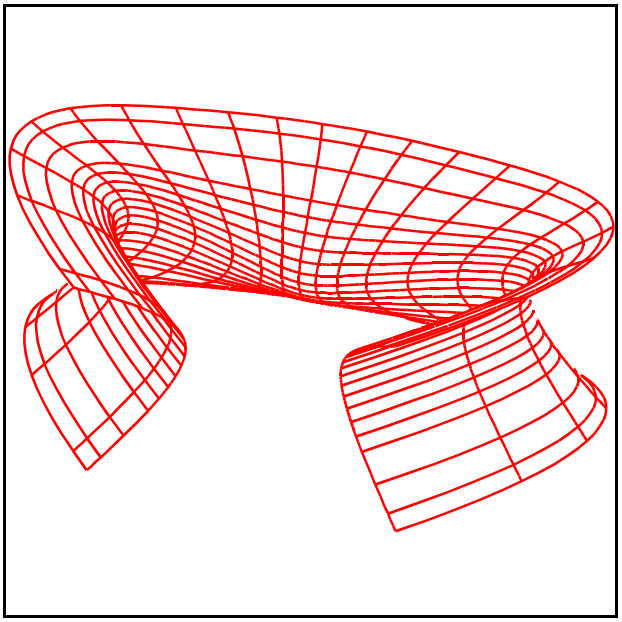


Figure 104 Edge derivatives

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Two edges can be forced to match, including their derivatives.

```
sd 1 hermite 4 3 2200
 2 0 0 0 2 0 -2 0 0 0 -2 0
 1 0 1 0 1 1 -1 0 1 0 -1 1
 2 0 2 0 2 2 -2 0 2 0 -2 2 ; ;
```

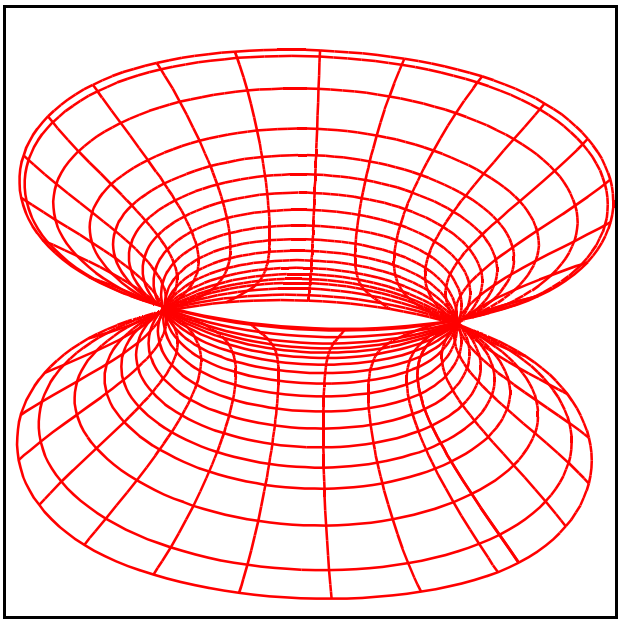


Figure 105 Periodic in 1 direction

Both pairs if edges can be matched to form a variation of a torus.

```
sd 1 hermite 4 3 2222
 2 0 0 0 2 0 -2 0 0 0 -2 0
 1 0 1 0 1 1 -1 0 1 0 -1 1
 2 0 2 0 2 2 -2 0 2 0 -2 2 ; ;
```

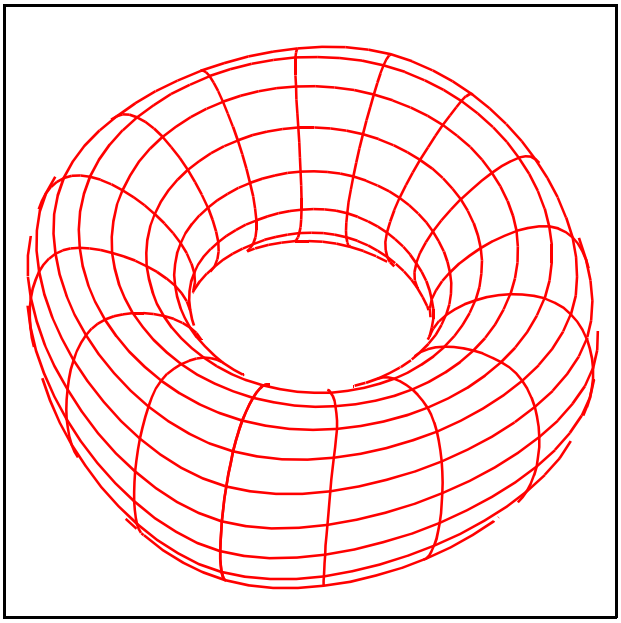


Figure 106 Periodic in both directions

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

igess import a parametric IGES surface

igess *seq_# trans* ;

where

seq_#	surface's sequence number in the IGES file
trans	list of transformations to be applied to it

Remarks

You make these surfaces with a CAD system, and write them to a file in IGES format. Then you can read the entity directory from the IGES file using the **igesfile** command (or less appropriately the **iges** command). The **igesfile** command assigns a sequence number to each of the parametric surfaces it encounters in the IGES file. The argument *seq_#* in this option is the parametric surface sequence number.

IGES surfaces have sequence numbers based on whether they are a NURBS, plane, or parametric type. This command imports one of the parametric type.

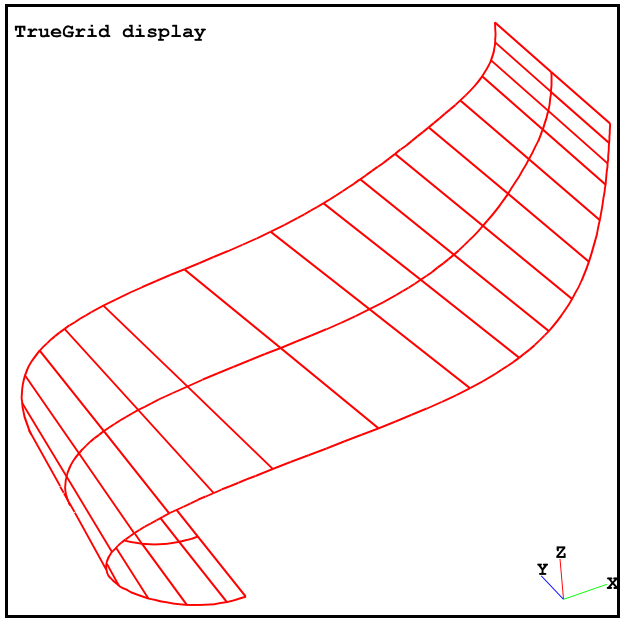


Figure 107 Imported parametric IGES Surface

When this IGES surface is assigned a surface definition number using this option, the surface data are read and the surface is rendered. The time it takes to render is a function of the curvature of the surface and of the trimming curves, because the surface must be evaluated more frequently in the areas of higher curvature.

When surfaces are trimmed, in which case all surfaces are usually trimmed, the surface will be counted twice, once as an untrimmed surface and once as a trimmed surface. All of the untrimmed surfaces are given sequence numbers first. Then the trimmed versions of the same surfaces are assigned sequence numbers in the same order as the untrimmed surfaces.

Alternatively, you can import all of the surfaces from an IGES file using the **iges** command. The **iges** command is much easier to use but you have to render all of the surfaces. You can use the **igessd** command to import a sequence of parametric surfaces instead of issuing the **sd** command many times.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

There may be times when you use the **iges** command to import all of the surfaces and then want to import a single surface with the **sd** command. You may wish to do this because you want the untrimmed version of the surface or you may want another copy of the surface with a transformation applied. The **igess** option of the **sd** command can also be applied to the parametric surfaces assigned sequence numbers from the **iges** command (i.e. there is no need to issue the **igesfile** command if you have already issued the **iges** command).

Example

```
igesfile surfaces.igs
```

The text window and the tsave file will show the sequence numbers for the surfaces found in the IGES file.

```
c  IGES file contained   114 surfaces from     1 to   114
c  IGES file contained    24 NURBS surfaces from     1 to    24
c  IGES file contained     4 planes from     1 to     4
```

The sequence numbers reported are then used to select the surface that is desired.

```
sd 1 igess 101;
```

If you read another IGES file, the sequence numbers are added to the existing sequence numbers.

```
igesfile more_surfaces.igs
```

The report to the text window and the tsave is:

```
c  IGES file contained    11 surfaces from   115 to   125
c  IGES file contained     4 NURBS surfaces from    25 to    28
c  IGES file contained     4 planes from     5 to     8

sd 2 igess 125 rx 30 rxy ;
```

In this last example, the surface is transformed by rotating about the x-axis by 30 degrees and reflected through the xy-coordinate plane before it is rendered.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

igesp import an IGES plane

igesp *seq_# trans* ;

where

seq_#	plane's sequence number in the IGES file
trans	list of transformations to be applied to it

Remarks

You make these surfaces with a CAD system, and write them to a file in IGES format. Then you can read the entity directory from the IGES file using the **igesfile** command (or less appropriately the **iges** command). The **igesfile** command assigns a sequence number to each of the plane surfaces it encounters in the IGES file. The argument *seq_#* in this option is the plane sequence number.

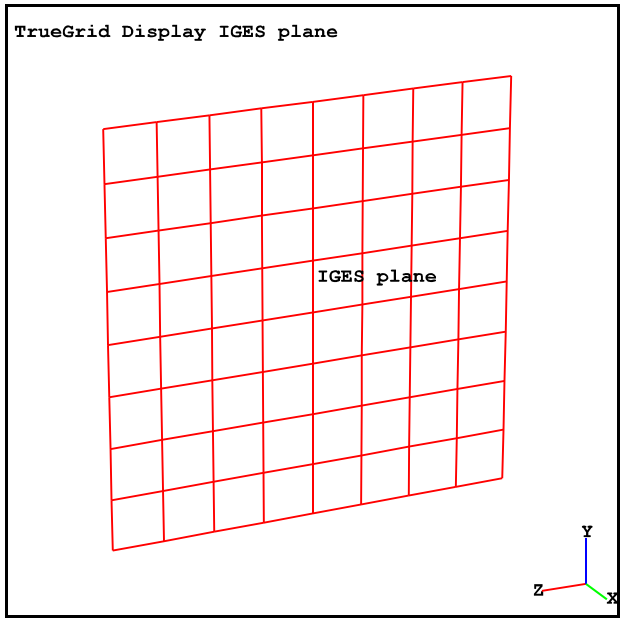


Figure 108 Imported IGES Plane

IGES surfaces have sequence numbers based on whether they are a NURBS, plane, or other type. This command imports one of the planar type.

When a planar surface is assigned a surface definition number using this option, the surface parameters are read and the surface is rendered. The time it takes to render is a function of the curvature of the trimming curves, because the surface must be evaluated more frequently in the high curvature areas.

When surfaces are trimmed, in which case all surfaces are usually trimmed, the surface will be counted twice, once as an untrimmed surface and once as a trimmed surface. All of the untrimmed surfaces are given sequence numbers first. Then the trimmed versions of the same surfaces are assigned sequence numbers in the same order as the untrimmed surfaces.

Alternatively, you can import all of the surfaces from an IGES file using the **iges** command. The **iges** command is much easier to use but you have to render all of the surfaces. You can use the **igespd** command to import a sequence of planar surfaces instead of issuing the **sd** command many times.

There may be times when you use the **iges** command to import all of the surfaces and then want to import a single surface with the **sd** command. You may wish to do this because you want the

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

untrimmed version of the surface or you may want another copy of the surface with a transformation applied. The **igesp** option of the **sd** command can also be applied to the planar surfaces assigned sequence numbers from the **iges** command (i.e. there is no need to issue the **igesfile** command if you have already issued the **iges** command).

Example

```
igesfile surfaces.igs
```

The text window and the tsave file will show the sequence numbers for the surfaces found in the IGES file.

```
c  IGES file contained   114 surfaces from     1 to   114
c  IGES file contained    24 NURBS surfaces from     1 to    24
c  IGES file contained     4 planes from     1 to     4
```

The sequence numbers reported are then used to select the surface that is desired.

```
sd 1 igesp 1;
```

If you read another IGES file, the sequence numbers are added to the existing sequence numbers.

```
igesfile more_surfaces.igs
```

The report to the text window and the tsave is:

```
c  IGES file contained    11 surfaces from   115 to   125
c  IGES file contained     4 NURBS surfaces from    25 to    28
c  IGES file contained     4 planes from     5 to     8
```

```
sd 2 igesp 6 rx 10 ry 10 ;
```

In this last example, the surface is transformed by rotating about the x-axis by 10 degrees and rotating about the y axis 10 degrees before it is rendered.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

intp interpolate a surface between two surfaces

intp sd_1 sd_2 *fraction*

Remarks

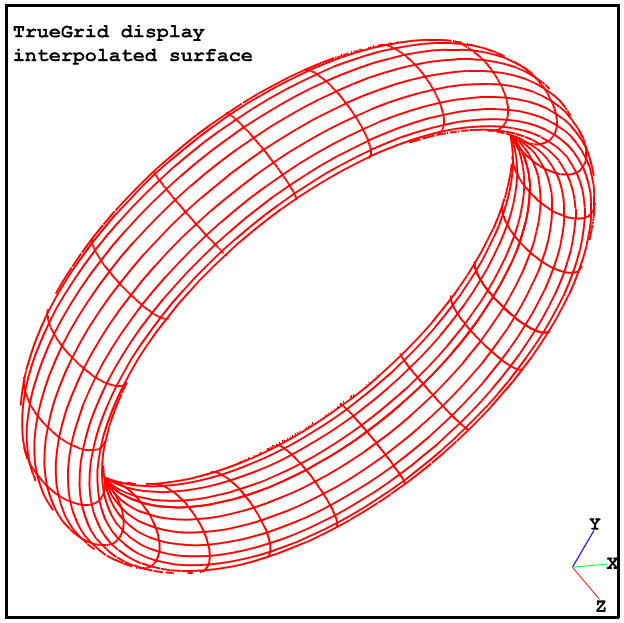


Figure 109 Surface Interpolated Between Two Other Surfaces

The two surfaces sd_1 and sd_2 must be defined before they are referenced by this command. The first surface is offset toward the second surface by projecting the points from the first surface onto the second surface. For the offset distance by specifying a fraction of the distance from the first surface to the second surface. You also specify the surfaces by giving their surface definition numbers (from the **sd** commands where they were defined) for the first two arguments. The first surface must be one of the following types: a rotated 2D curve (**cr**, **crx**, **cry**, or **crz**), a sphere (**sp**), a rotated ellipse (**er**), a structured mesh (**mesh**), a linear interpolation between two curves (**rule2d**), an interpolation between 2 3D curves (**rule3d)**, a torus (**tr**), 2D cross sections placed along a 2D curve (**swept**), another **intp**, or IGES surfaces other than a plane. This feature has limited use and you should take care in creating an **intp** surface. This is the only surface type valid for the variable shell surface command **ssf**.

Example

```
sd 1 ts 0 0 0 0 0 1 5 0 1
sd 2 cy 0 0 0 0 0 1 5
sd 3 intp 1 2 .5
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

iplan infinite plane defined by an implicit function

iplan *a b c d*

where the arguments are coefficients in the equation below.

Remarks

This is an infinite surface. The graphics will only show a portion of the surface. That portion shown in the graphics changes as the objects in the picture change. The plane is defined by the

$$a\,x + b\,y + c\,z = d$$

set of points that satisfy
where the coordinate triplet *(x,y,z)* is any point on the plane and where at least one of the variables *a*, *b*, or *c* is different from zero as shown in **Figure 110**.

```
sd 3 iplan 1 -1 1 1
```

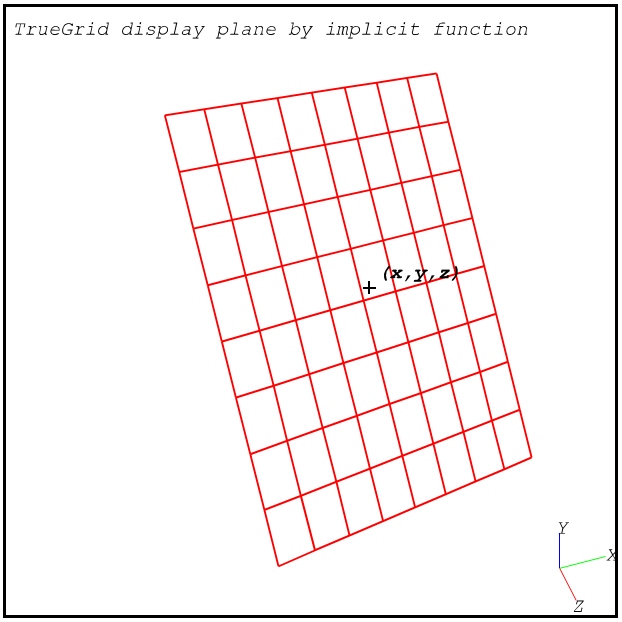


Figure 110 Plane specified by an implicit function

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

mesh surface by tabular points

mesh *m n* x_{11} y_{11} z_{11} x_{21} y_{21} z_{21} ... x_{mn} y_{mn} z_{mn} *trans ;*

where

m	number of rows in the table or 2D array
n	number of columns in the table or 2D array
(x_{ij}, y_{ij}, z_{ij})	3D coordinates at position (i,j) in the table or 2D array
trans	standard coordinate transformation (sequence of operators)

Remarks

Mesh defines a surface by specifying a table or 2D array of points to be part of the surface. The rest of the surface is implicitly defined by bilinear interpolation between those points. This type of surface provides a general way to use experimental or computational data from any source.

The first two integers, *m* and *n*, give the number of points in the rows and columns, respectively. These parameters must be greater than 1. The remaining arguments give the positions of each mesh point in physical space. These points should be ordered the way Fortran would order a 2D array with *m* rows and *n* columns, i.e. with:

(i,j) = (1,1), (2,1), ... , (*m*,1), (1,2), ... , (*m*,2), ... , (1,*n*) , ... , (*m*,*n*).

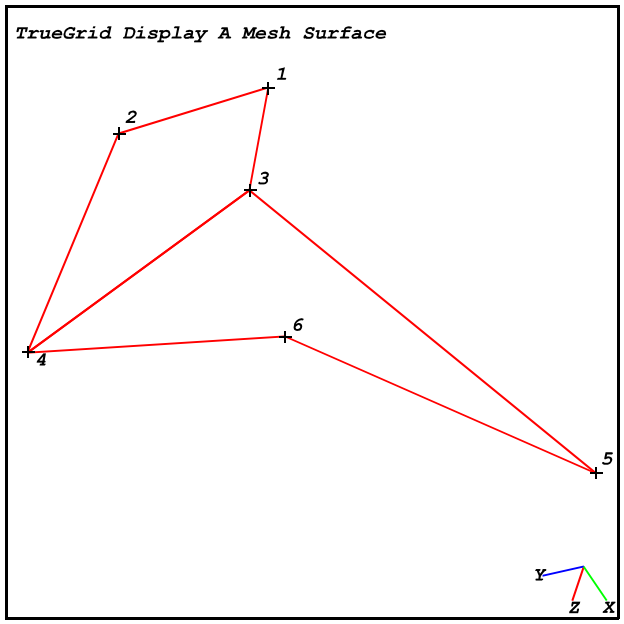


Figure 111 2 by 3 Mesh Surface

Example

```
sd 1 mesh 2 3
1.125 0.999 1.269
1.566 1.875 0.869
2.092 1.753 0.644
2.087 2.377 1.170
3.053 0.936 1.119
2.606 1.891 0.712 ;
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

nrbs NURBS

nrbs *i-degree j-degree*
iknot$_1$ ... *iknot*$_{k1}$; *jknot*$_1$ .... *jknot*$_{k2}$;
weight$_{1,1}$ *weight*$_{1,1}$ *weight*$_{1,1}$ ... *weight*$_{1,n2}$ *weight*$_{1,n2}$ *weight*$_{1,n2}$
...
weight$_{n1,1}$ *weight*$_{n1,1}$ *weight*$_{n1,1}$ ... *weight*$_{n1,n2}$ *weight*$_{n1,n2}$ *weight*$_{n1,n2}$;
xcontrol$_{1,1}$ *ycontrol*$_{1,1}$ *zcontrol*$_{1,1}$ ... *xcontrol*$_{1,n2}$ *ycontrol*$_{1,n2}$ *zcontrol*$_{1,n2}$
...
xcontrol$_{n1,1}$ *ycontrol*$_{n1,1}$ *zcontrol*$_{n1,1}$ ... *xcontrol*$_{n1,n2}$ *ycontrol*$_{n1,n2}$ *zcontrol*$_{n1,n2}$;
trans ;
where

i-degree j-degree	degrees of the surface polynomial in the i and j direction.
i*knot*$_1$ ... *iknot*$_{k1}$	parametric coordinates of *k1* knots (0..) in the i direction.
j*knot*$_1$ ... *jknot*$_{k2}$	parametric coordinates of *k2* knots (0..) in the j direction.
weight$_{1,1}$ *weight*$_{1,1}$ *weight*$_{1,1}$ ... *weight*$_{1,n2}$ *weight*$_{1,n2}$ *weight*$_{1,n2}$ *weight*$_{n1,1}$ *weight*$_{n1,1}$ *weight*$_{n1,1}$ ... *weight*$_{n1,n2}$ *weight*$_{n1,n2}$ *weight*$_{n1,n2}$;	weights for each control point
xcontrol$_{1,1}$ *ycontrol*$_{1,1}$ *zcontrol*$_{1,1}$ ... *xcontrol*$_{1,n2}$ *ycontrol*$_{1,n2}$ *zcontrol*$_{1,n2}$ *xcontrol*$_{n1,1}$ *ycontrol*$_{n1,1}$ *zcontrol*$_{n1,1}$ ... *xcontrol*$_{n1,n2}$ *ycontrol*$_{n1,n2}$ *zcontrol*$_{n1,n2}$;	triplets of coordinates of control points
trans	sequence of simple operators to transform the coordinates

where
$n_1 = k_1$ - *multiplicity*$_1$
$n_2 = k_2$ - *multiplicity*$_2$

Remarks

First, a few notes on the parameters. The multiple knots have to be specified for the starting and ending point of the curve. The number of multiple knots is called multiplicity. The multiplicity for the starting and ending point determines the order of the curve. (2 - linear, 3 - quadratic, 4 - cubic ...).

multiplicity = degree + 1

The multiplicity has to be the same for the starting and ending points.

A normal offset can be applied to this surface by issuing the normal option followed by the offset as the first operator in the transformation *trans*.

NURBS surfaces are also available through the IGES interface.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

The NURBS surface (**Figure 112**) is defined by the same control points as the B-Spline Surface (**Figure 81**). The weights are specified for each control point. Every weight is 1.0 except 4 control points with weights 10.0 (**Figure 112**). The NURBS surface is passing closer to the control points with w=10.

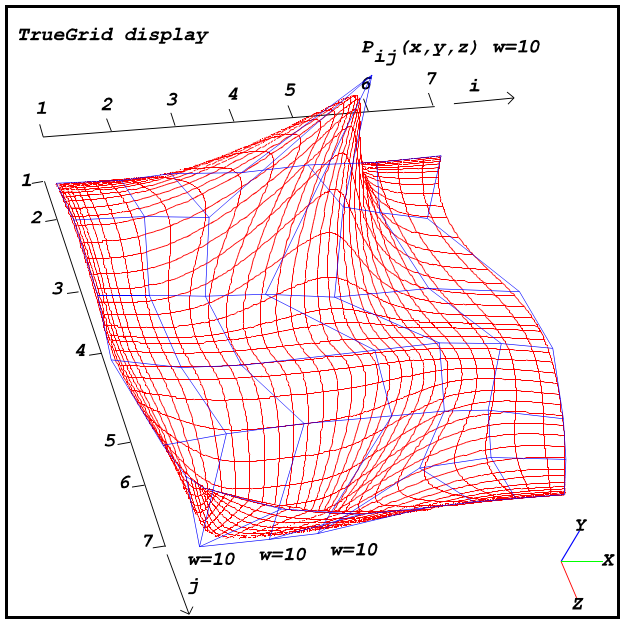


Figure 112 NURBS Surface

```
sd 1 nrbs 3 3
0 0 0 0 1 2 3 4 4 4 4;
0 0 0 0 1 2 3 4 4 4 4;
1 1 1 1 1 1 1
1 1 1 10 1 1 1
1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 10 10 10 1 1 1
1 1 1 1 1 1 1;
...                    c x,y,z coordinates of control points
...                    c organized into 7 rows with 7 columns
...                    c the same as for BSPS surface
```

nurbs — import a NURBS surface

nurbs *seq_# trans* ;

where

seq_#	NURBS's surface sequence number in the IGES file
trans	list of transformations to be applied to it

Remarks

You make these surfaces with a CAD system, and write them to a file in IGES format. Then

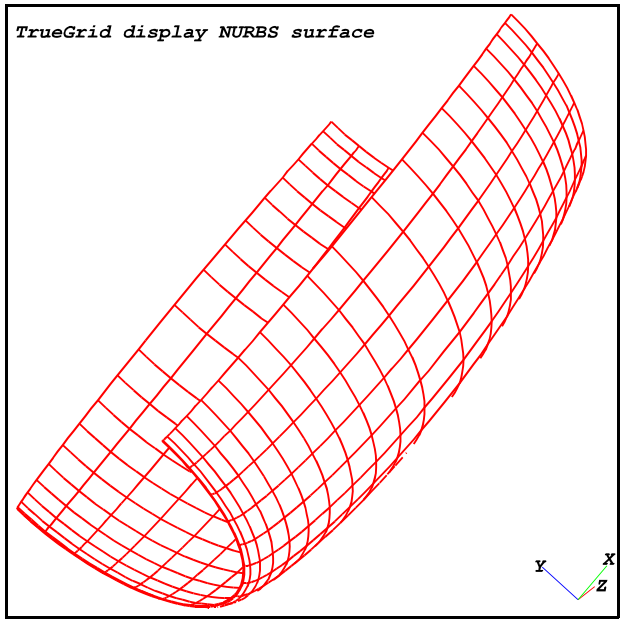


Figure 113 NURBS Surface from an IGES File

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

you can read the entity directory from the IGES file using the **igesfile** command (or less appropriately the **iges** command). The **igesfile** command assigns a sequence number to each of the NURBS surfaces it encounters in the IGES file. The argument *seq_#* in this option is the NURBS sequence number.

IGES surfaces have sequence numbers based on whether they are a NURBS, plane, or parametric type. This option imports one of the NURBS type.

When a NURBS surface is assigned a surface definition number using this option, the surface parameters are read and the surface is rendered. The time it takes to render is a function of the curvature of the surface and the curvature of the trimming curves, because the surface must be evaluated more frequently in the high curvature areas.

When surfaces are trimmed, in which case all surfaces are usually trimmed, the surface will be counted twice, once as an untrimmed surface and once as a trimmed surface. All of the untrimmed surfaces are given sequence numbers first. Then the trimmed versions of the same surfaces are assigned sequence numbers in the same order as the untrimmed surfaces.

Alternatively, you can import all of the surfaces from an IGES file using the **iges** command. The **iges** command is much easier to use but you have to render all of the surfaces. You can use the **nurbsd** command to import a sequence of NURBS surfaces instead of issuing the **sd** command many times.

There may be times when you use the **iges** command to import all of the surfaces and then want to import a single surface with the **sd** command. You may wish to do this because you want the untrimmed version of the surface or you may want another copy of the surface with a transformation applied. The **nurbs** option of the **sd** command can also be applied to the NURBS surfaces assigned sequence numbers from the **iges** command (i.e. there is no need to issue the **igesfile** command if you have already issued the **iges** command).

Example

```
igesfile surfaces.igs
```

The text window and the tsave file will show the sequence numbers for the surfaces found in the IGES file.

```
c  IGES file contained   114 surfaces from     1 to   114
c  IGES file contained    24 NURBS surfaces from     1 to    24
c  IGES file contained     4 planes from     1 to     4
```

The sequence numbers reported are then used to select the surface that is desired.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
sd 1 nurbs 16;
```

If you read another IGES file, the sequence numbers are added to the existing sequence numbers.

```
igesfile more_surfaces.igs
```

The report to the text window and the tsave is:

```
c  IGES file contained    11 surfaces from   115 to   125
c  IGES file contained     4 NURBS surfaces from    25 to    28
c  IGES file contained     4 planes from     5 to     8

sd 2 nurbs 27 rz 90 mx .1 ;
```

In this last example, the surface is transformed by rotating about the z-axis by 90 degrees and translating in the x direction .1 before it is rendered.

pipe sweep a pipe shape along an arbitrary 3D curve

pipe *3d_curve* $radius_1$ arc_length_1 $radius_2$ arc_length_2 ... ; *trans* ;

where

3d_curve	3D curve that is used to sweep along
$radius_i$ arc_length_i	radius and arc length pairs specified along the 3D curve
trans	optional transformation at the end

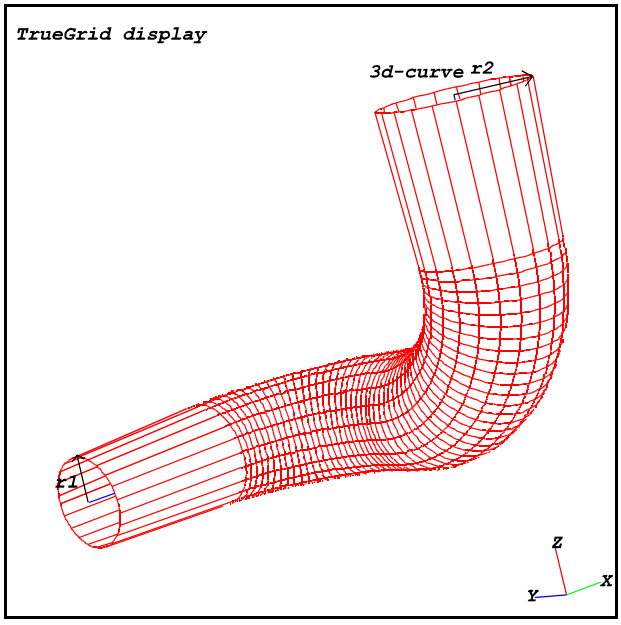


Figure 114 pipe surface

Remarks

This command allows you to define a pipe with

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

varying cross-section. The arc length varies from a value of 0 at one end of the 3D curve to a value of 1 at the other end. In other words, the arc length is the fractional value of the entire 3D-curve length at the point where the circular cross-sectional radius is defined. You may specify as many cross-sections as you like.

Example

```
curd 1 lp3 -1 1 1 0 1 1;;
3dfunc 0 90 cos(u;sin(u);1;;
3dfunc 180 270 1;sin(u);cos(u)+2;;
lp3 1 -1 3; ;
sd 1 pipe 1 .3 0 .5 1;;
```

pl2 plane specified by two points

pl2 *system point system point*

where

system		one of the three coordinate systems:
	rt	for Cartesian
	sp	for spherical
	cy	for cylindrical
point		set of 3 coordinate in the appropriate coordinate system

Remarks

This option defines the plane using 2 points. The first point is on the plane, the second point in not on the plane and on the line normal to the plane through the first point.

Example

```
sd 1 pl2 rt 1 1 1 rt 2 2 2 ;
```

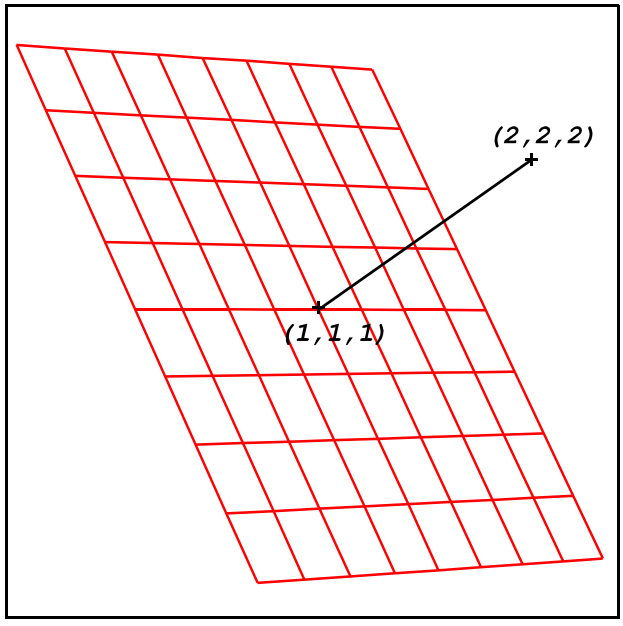


Figure 115 Plane by 2 points

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

pl3 plane specified by three points

pl3 *system point system point system point*

where

system		one of the three coordinate systems:
	rt	for Cartesian
	sp	for spherical
	cy	for cylindrical
point		set of 3 coordinate in the appropriate coordinate system

Remarks

In Cartesian coordinates, the point is designated by its x, y, and z coordinate values. In cylindrical coordinates, the radius is followed by the angle and z-coordinates. In spherical coordinates, the radius is first, followed by the polar and azimuthal angles. All angles are in degrees. Remember, the three points must not be co-linear. This is an infinite surface. The graphics will only show a portion of the surface. That portion shown in the graphics changes as the objects in the picture change.

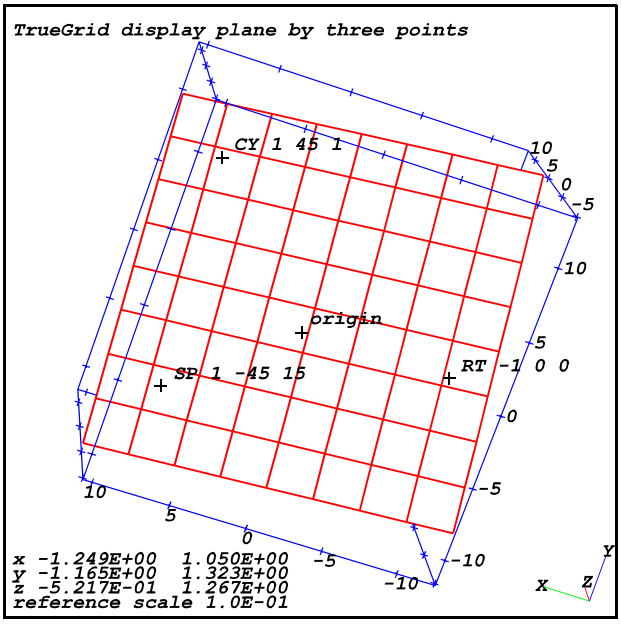


Figure 116 Plane specified by 3 points

Example

```
sd 1 pl3 cy 1 45 1 sp 1 -45 15
rt -1 0 0
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

pl3o plane specified by 3 points and an offset

pl3o *system point system point system point offset*

where

system	one of the three coordinate systems:
rt	for Cartesian
sp	for spherical
cy	for cylindrical
point	set of 3 coordinate in the appropriate coordinate system
offset	normal offset distance

Remarks

This command defines a plane specified by three points and a positive normal offset. The direction of the normal is defined by the right hand rule through the 3 points.

The point is designated by its x, y, and z coordinate values. In cylindrical coordinates, the radius is followed by the angle and z-coordinates. In spherical coordinates, the radius is first, followed by the polar and azimuthal angles. All angles are in degrees.

This is an infinite surface. The graphics will only show a portion of the surface. That portion shown in the graphics changes as the objects in the picture change.

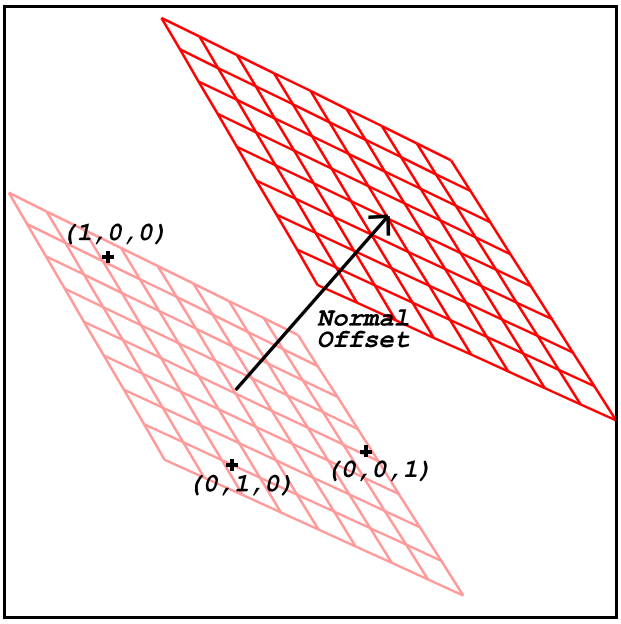


Figure 117 Normal offset plane

Example

```
sd 1 pl3o rt 1 0 0 rt 0 1 0 rt 0 0 1 1 ;
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

plan infinite plane

plan $x_0\ y_0\ z_0\ x_n\ y_n\ z_n$

where

x_0	first coordinate of a point on the axis of rotation
y_0	second coordinate of a point on the axis of rotation
z_0	third coordinate of a point on the axis of rotation
x_n	first component of the normal vector
y_n	second component of the normal vector
z_n	third component of the normal vector

Remarks

A plane containing the point (x_0,y_0,z_0) and normal to the vector (x_n,y_n,z_n) as shown in **Figure 118**.

```
sd 1 plan 0 0 0 1 0 0
```

Remarks

This is an infinite surface. The graphics will only show a portion of the surface. That portion shown in the graphics changes as the objects in the picture change.

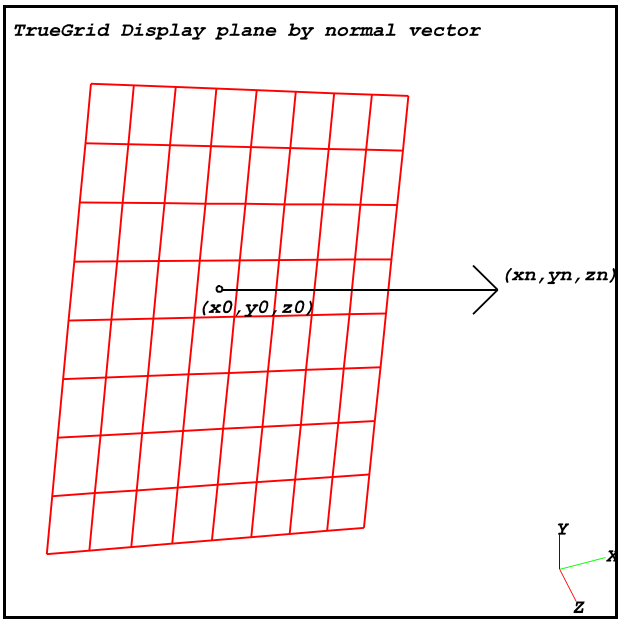


Figure 118 Plane by a point and a normal vector

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

poly convert a polygon set into a surface

poly *polygon_set trans ;*

where

polygon_set	named set of polygons
trans	sequence of simple operators to transform the coordinates

Remarks

You can create a polygon surface using the **pset** command and the interactive feature to select or modify the polygons in a set with the ***Sets*** window. The ***Sets*** window can be activated in the merge phase from the ***Pick*** panel in the Environment Window. These features, along with the **wrsd** command, can be used to sort out complex polygon surfaces and split them into multiple surfaces or remove features. This can also be used to create a normal offset by using the normal option as one of the transformation primitives.

Use the **fetol** command before this command to extract interior features as edges.

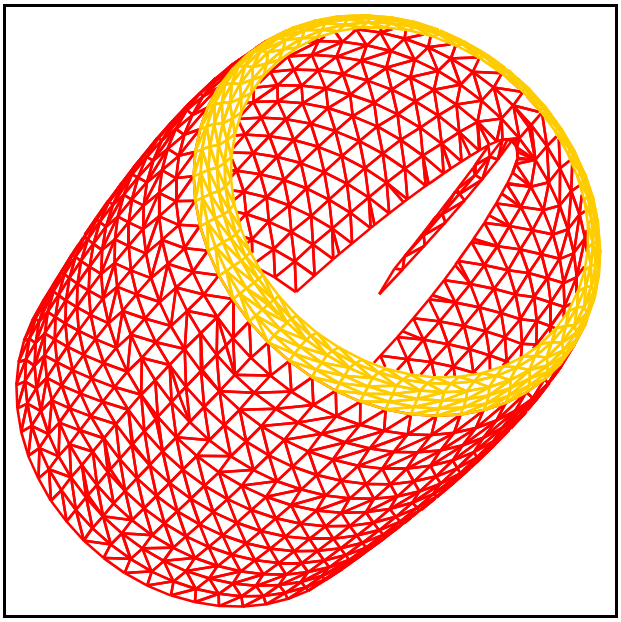


Figure 119 Surface from a polygon set

Example

It is assumed in this example that a polygon set was formed using the Sets window in the Pick panel of the Environment window. This polygon set was named p1. Then the command to create a surface form this set is:

```
sd 2 poly p1;
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

pr paraboloid (parabola revolved about an axis)

pr $x_0\ y_0\ z_0\ x_n\ y_n\ z_n\ r_1\ t_1\ r_2\ t_2\ r_3\ t_3$

where

x_0	first coordinate of a point on the axis of rotation
y_0	second coordinate of a point on the axis of rotation
z_0	third coordinate of a point on the axis of rotation
x_n	first component of axis direction vector
y_n	second component of axis direction vector
z_n	third component of axis direction vector
r_1	radius at the cross section along axis at t_1
t_1	position along axis with radius r_1
r_2	radius at the cross section along axis at t_2
t_2	position along axis with radius r_2
r_3	radius at the cross section along axis at t_3
t_3	position along axis with radius r_3

Remarks

A parabola is rotated about an axis given by (x_0,y_0,z_0) and (x_n,y_n,z_n). You define the parabola by specifying three points that it passes through in any planar cross section containing the axis of symmetry: (r_1,t_1), (r_2,t_2), and (r_3,t_3). The t coordinate is measured along the axis, starting at the point (x_0,y_0,z_0). The r coordinate is measured perpendicular to the axis. The three points can be in any order. These points must uniquely specify a parabola. This means that the coordinates must satisfy the condition::

$$|t_1^2\times(t_3-t_2)+t_2^2\times(t_1-t_3)+t_3^2\times(t_2-t_1)| > 10^{-18}$$

A point projected to this surface should not be on the axis of symmetry. See the example in **Figure 120**. This is an infinite surface. The graphics will only show a portion of the surface. That portion shown in the graphics changes as the objects in the picture change.

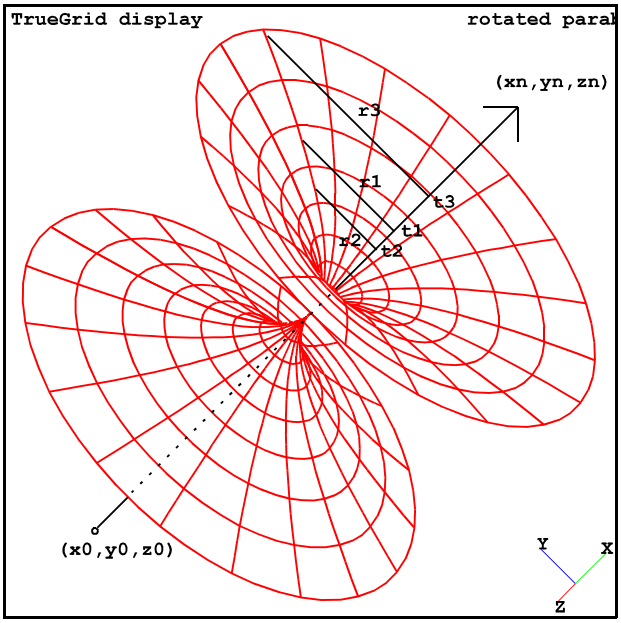


Figure 120 Parabola specified by three points

Example

```
sd 4 pr 0 0 0 1 0 0 .75 8 1.2675
8.3 3 9
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

r3dc 3D curve revolved about an axis

r3dc $x_0\ y_0\ z_0\ x_n\ y_n\ z_n$ *3D_curve begin_angle end_angle tran ;*

where

(x_0,y_0,z_0)	point on the axis of rotation
(x_n,y_n,z_n)	vector parallel to the axis of rotation
3D_curve	3D curve being rotated (generatrix)
begin_angle	beginning angle for the rotational sweep
end_angle	ending angle for the rotational sweep
trans	sequence of simple operators to transform the coordinates

Remarks

This command forms the surface of rotation determined by the generatrix, *3D_curve*, the axis of rotation (defined by the point (x_0,y_0,z_0) and the direction vector (x_n,y_n,z_n)) and the starting and ending angles (*begin_angle* and *end_angle*, respectively).

The surface is created by rotating the generatrix around the axis of rotation from the beginning angle to the ending angle. For any point on the generatrix, the rotational arc created lies on the circle centered along the axis rotation, orthogonal to (x_n,y_n,z_n) and through that point on the generatrix. The portion of the circle retained is measured counter-clockwise starting *begin_angle* degrees from the generatrix point to *end_angle*.

The best results occur when the generatrix is coplanar with the axis, and the worst results occur along segment tangent to the generated arc.

To improve results add a point to the generatrix near to the tangent points. The geometry tolerance (set with **getol**) may need to be reduced so that the line thinning algorithm does not remove new points.

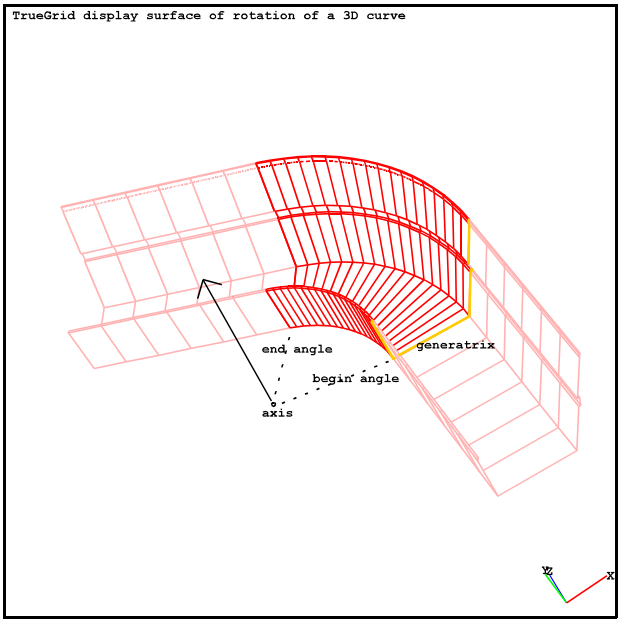


Figure 121 Surface of revolution of a 3D curve

Example

```
curd 1 lp3 6.01 0 3.35  6.1 0
3.5 6 0 3.5 6 0 .5  10 0 0  11 0
2.5 11.1 0 2.5  11.2 0 3 11.1 0
3 12.1 0 5.5  12.05 0 5.5 11.995
0 5.25;;
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
sd 1 r3dc 0 0 0 0 0 1 1 0 60;
curd 2 cpcd 1 my -10 mz .1;
sd 2 rule3d 1 2;;
curd 3 cpcd 1 rz 60;
curd 4 cpcd 3 mx -8.66 my 5 mz -.1;
sd 3 rule3d 3 4;;
```

rule2d ruled surface between two 2D curves

rule2d y_1 ln_1 y_2 ln_2 *trans* ;

where

y_1	y-coordinate of the plane for the first 2D curve
ln_1	2D curve ID which will form one edge of the surface
y_2	y-coordinate of the plane for the second 2D curve
ln_2	2D curve ID which will form the opposite edge
trans	sequence of simple operators to transform the coordinates

Remarks

This surface is formed by a linear interpolation between two 2D curves. Prior to using this command, you must have defined the 2D curves numbered ln_1 and ln_2, using commands like **ld**. These 2D curves would originally lie in the plane y=0 (the xz-plane), but **rule2d** will translate them in the y-direction to the y=*y1* and y=*y2* planes, respectively. The interpolation amounts to constructing a straight line from each point of one translated curve to the corresponding point of the other translated curve. The order of points along the two curves (orientation) must be the same. To reverse the order of the points in one of the curves, specify that 2D curve definition number as negative. The points that correspond to each other are those which have the same relative arc length position on their respective curves. At least one of the two 2D curves must have positive arc length.

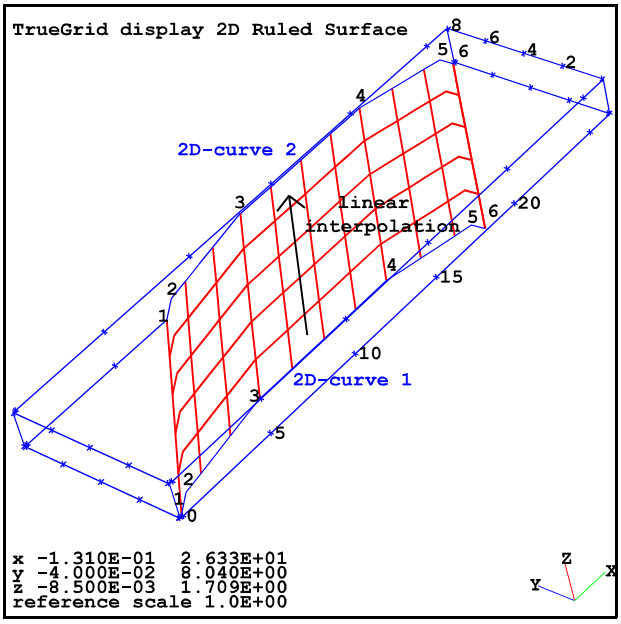


Figure 122 Ruled Surface From Two 2D Curves

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
ld 1 lp 0 0  .5 .7  5 1.7 12.5 1.7 17.5 .7 18.2 0;
ld 2 lp 8 0 8.5 .7 13 1.7 20.5 1.7 25.5 .7 26.2 0;
sd 1 rule2d 0 1 8 2;
```

rule3d ruled surface between two 3D curves

rule3d *3D-curve$_1$ 3D-curve$_2$ trans ;*

where

3D-curve$_1$	first boundary 3D curve
3D-curve$_2$	second boundary 3D curve
trans	sequence of simple operators to transform the coordinates

Remarks

A surface is formed by a linear interpolation in Cartesian coordinates between two 3D curves. At least one of the two curves must have a positive arc length. Both curves must be already defined. 3D curves are defined with the **curd** command or read from an **IGES** file. The direction of the curves is critical to the interpolation of the surface. Check the order of the points in each of the curves by displaying the points with the **label** command. In order to reverse the order of a curve, specify the curve definition number with a negative number. The surface is constructed by pairing each point on one curve with a point on the other curve. These two points are connected by a line segment. Points along the two curves are paired by relative arc length. The following is an example of three 3D curves, which are annotated below, and the two surfaces interpolated between pairs of curves.

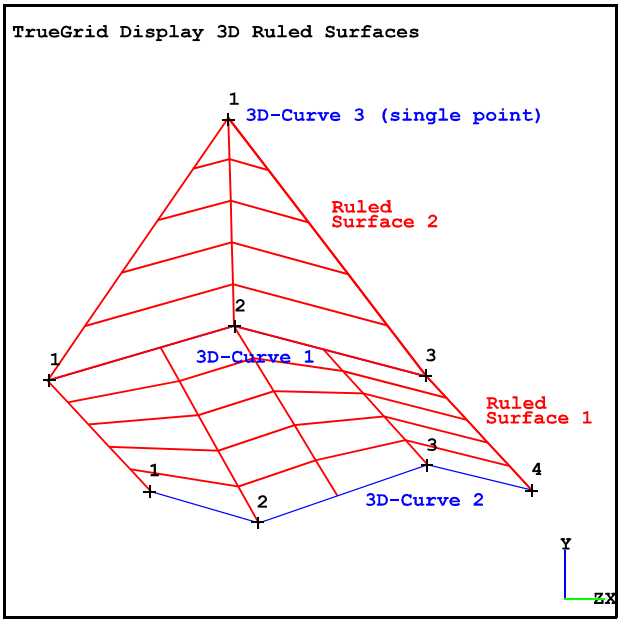


Figure 123 3D Ruled Surfaces From 3D Curves

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
curd 1 lp3 1 2 1 3 2.5 1.1 5 2 1.5;;
curd 2 lp3 1.5 1 2 2 .75 3 3.5 1.25 3.5 4.5 1 3.75;;
curd 3 lp3 3 4.5 1;;
sd 1 rule3d 1 2;
sd 2 rule3d 1 3;
```

sds union of surfaces

sds $s_1\ s_2\ \ldots\ s_n$;

where s_i is the number of a surface previously defined

Remarks

The surfaces that form this union do not need to meet perfectly. There can be large gaps and overlappings.

This is the most important surface type when dealing with CAD models. Typically, the CAD model is broken into many surfaces for construction purposes. No consideration is made in the construction of the surfaces for mesh generation. In most cases, it is more convenient, if not essential, to form a mesh across many of these surfaces. Use this option to form a composite surface and it will be as if it were one surface. The projection of a face of the mesh works the same for both single and composite surfaces.

Sometimes, this surface type can be used to group surfaces when there are many surfaces to sort (i.e. it is very useful when working with IGES input files).

When a node is projected to a multiple surface, it is projected to each of the component surfaces. The point of projection which is the shortest distance among all of the component surface projections is defined as the projection point for the multiple surface. This makes it possible for the surfaces to overlap or not meet perfectly. A closest point will always be found.

A multiple surface list can include surfaces defined with the **sds** option.

An effective way to prepare a list of surfaces is to first display all surface definitions (**dasd**). Then, in the Environment Window, use the ***Display List***, ***Surface*** button in the Apply Action To: menu, and the ***Remove*** button to remove those surfaces which you do not want in your grouped surface. When you only have those surfaces displayed which you wish to group together, use the **lasd** command to list all of the displayed surfaces. Type

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

sd *some_number* sds

in the text window and cut and paste the listed surfaces, followed by a semi-colon. (Where *some_number* is the definition number of the surface that you are creating.)

Care is needed when using this option to define several multiple surfaces, each containing some of the same component surfaces. The intersection of the two multiple surfaces includes the surfaces that they both have in common. This would be an extreme case of two surfaces being tangent. This can be an effective technique when you want an edge of the mesh to lay along the middle of a surface common to both composites.

Example

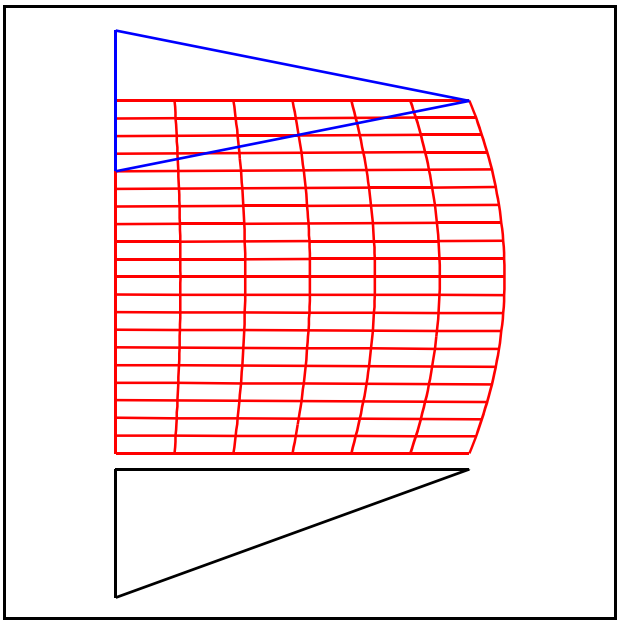


Figure 124 Union of 3 surfaces with errors

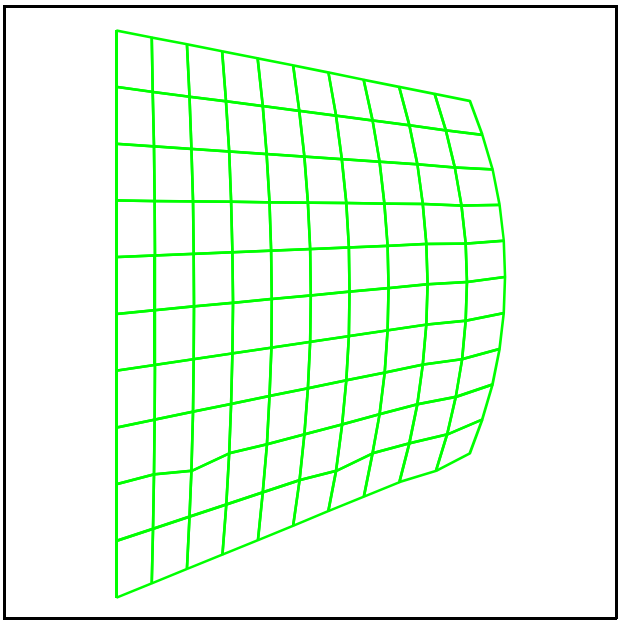


Figure 125 Mesh projected to union of surfaces

Typically, a CAD model will have inaccuracies in the geometry such as gaps between surfaces or surfaces that overlap. This example is an exaggeration of this common problem to demonstrate the robust nature of the **sds** option and the projection method. These two features go hand in hand. In this example, three surfaces form a composite surface. There is a gap between two of the surfaces. There is an overlap between two surfaces. When a face of the mesh is projected to this union of surfaces, the only effect these inaccuracies have on the mesh is that some of the interior nodes are perturbed because they cannot project into the gap. This is usually not a problem because the gaps

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

are usually not that large.

In this next example, two surfaces form a composite surface. The **sds** option is a natural way to combine tangent surfaces so that you can avoid intersecting tangent surfaces.

```
sd 1 function 0 90 0 360
cos(u)*cos(v);
cos(u)*sin(v);
3*sin(u);;
sd 2 function -90 0 0 360
cos(u)*cos(v);
cos(u)*sin(v);
.5*sin(u);;
sd 3 sds 1 2;
```

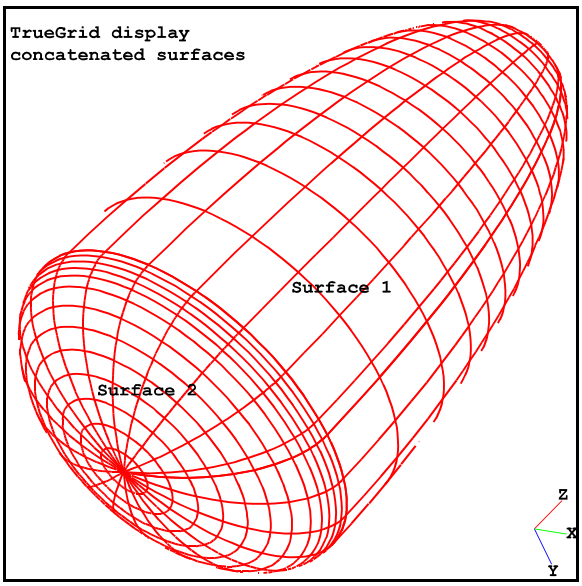


Figure 126 Two surfaces concatenated

sp sphere

sp x_0 y_0 z_0 *radius*

where

x_0	first coordinate of the center of the sphere
y_0	second coordinate of the center of the sphere
z_0	third coordinate of the center of the sphere
radius	radius of the sphere

Remarks

This command defines a sphere with center (x_0,y_0,z_0) and radius *radius*, as shown in the following picture. The radius must be positive. When projecting a point onto the sphere, be sure that the point is away from the center of sphere. Projections are made along rays emanating from the center of the sphere.

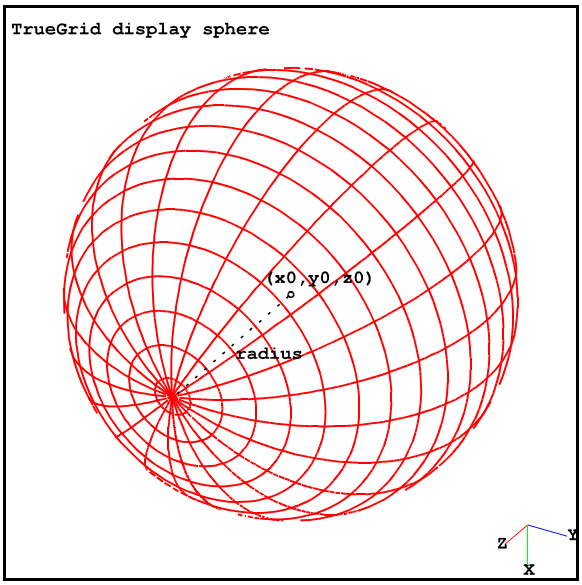


Figure 127 Sphere specified by center and radius

Example

```
sd 10 sp 1 0 -1 9
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

stl read the standard ASCII STL file

stl *file trans* ;

where

file path and file name (case sensitive in UNIX/LINUX)

trans sequence of simple operators to transform the coordinates

Remarks

This command permits you to import triangular surfaces via StereoLithography (STL) files. Both the ASCII and binary versions of the file are supported.

Use the **fetol** command before this command to extract interior features as edges.

Example

```
fetol 110
sd 1 stl lcab2b.stl ;
```

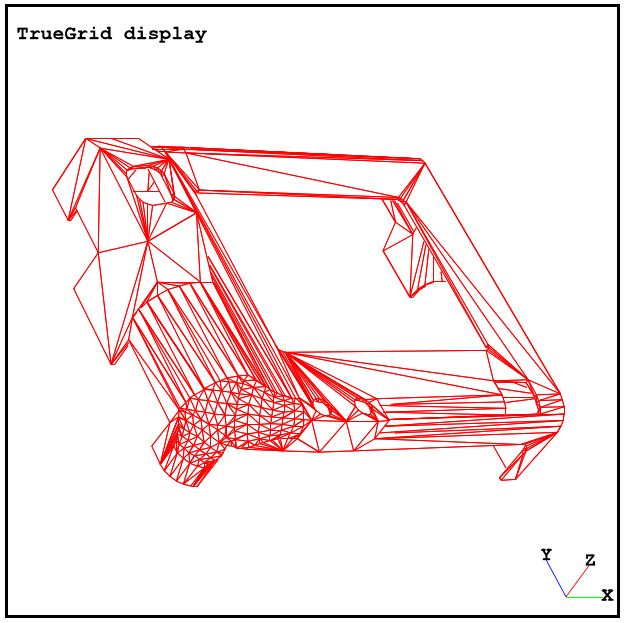


Figure 128 Surface imported from stl file

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

swept sweep 2D curves along a 2D curve

swept ln_0 *direction* ln_1 α_1 ... ln_n α_n ; *trans* ;

where

ln_0	2D curve number
direction	either **r** for the right side or **l** for the left side
ln_i α_i	pairs of 2D curve numbers and relative arc lengths
trans	sequence of simple operators to transform the coordinates

Remarks

This is a surface which exactly matches various 2D curves at an arbitrary number of cross-sections.

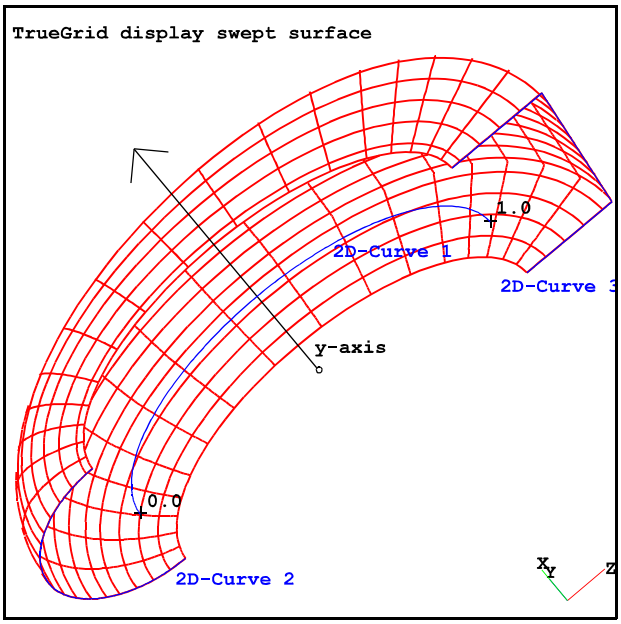


Figure 129 Swept surface with 2 cross sections

The 2D curve ln_0 is used as the base of the surface. This 2D curve is on the xz-plane. Positions are measured along this curve by relative arc length. The first point on the base curve has a relative arc length of zero. The last point on the base curve has a relative arc length of 1.0. A cross section of the surface can be specified at any point along the base curve by specifying the relative arc length and the 2D curve which forms the cross section at that point. The 2D curves ln_1, ... ln_n form the cross sections. The parameters $\alpha_1 \ldots \alpha_n$ are the relative arc lengths of the cross sections, respectively. These arc length parameters must all be between 0 and 1, inclusive. The plane containing the 2D cross section curve will be placed normal to the curve such that the origin in the 2D cross section plane matches the selected point on the base curve. If the *direction* is **r** for right, then the right side of the base curve is the positive direction in the cross section. If *direction* is specified as **l** for left, then the left side of the base curve is the positive direction for the planar cross section. The 2D cross section curves must be specified in order along the base curve. The surface between the cross sections is constructed by linearly interpolating the specified cross sections. The ordering of the points in the 2D cross section curves must match. The arc length of all of the 2D curves must be positive. Care must be taken when using this surface because it is easy to specify cross sections that cause the surface to overlap itself when the base curve has a lot of curvature. All 2D curves must be defined before they are referenced by this command. A transformation can be applied to this surface.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

In this example the base curve is half of an ellipse. The first cross section is a half of a circle. The last cross section is a half of a cube. The surface transitions smoothly from the smooth arc to 90 degree corners.

```
ld 1 lep 1 2 1 2 -90 90 0;
ld 2 lep 1 1 0 0 -90 90 0;
ld 3 lp2 0 -1 1 -1 1 1 0 1;;
sd 1 swept 1 r 2 0 3 1;;
```

ts torus

ts $x_0\ y_0\ z_0\ x_n\ y_n\ z_n\ r_1\ t\ r_2$

where

x_0	first coordinate of a point on the axis of rotation
y_0	second coordinate of a point on the axis of rotation
z_0	third coordinate of a point on the axis of rotation
x_n	first component of axis direction vector
y_n	second component of axis direction vector
z_n	third component of axis direction vector
r_1	larger radius
t	distance from the pont $(x_0\ y_0\ z_0)$ to the center of the torus
r_2	cross section radius

Remarks

(x_0,y_0,z_0) and (x_n,y_n,z_n) give the axis of symmetry, which passes through the center of the hole in the middle. t is the distance (along the axis) from (x_0,y_0,z_0) to the center point of the torus. The torus has major radius r_1 and minor radius r_2, and both must be positive. A point projected onto this surface should be away from the axis of symmetry and away from the circle at the center of the interior of the torus.

Example

```
sd 4 ts 0 0 0 1 0 1 4 6 1
```

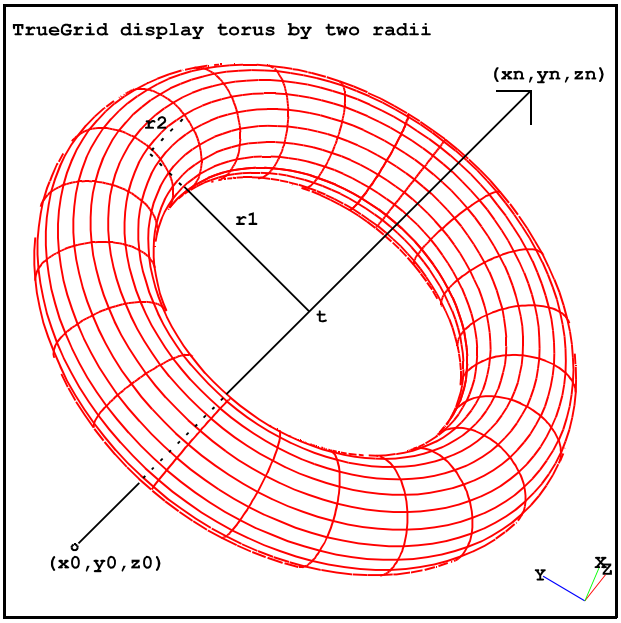


Figure 130 Torus specified by two radii

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

xcy to transform an infinite x-axis cylinder

xcy *radius trans* ;

where

radius	is the radius of the cylinder, and
trans	sequence of simple operators to transform the coordinates

Remarks

An infinite cylinder starts out with its axis aligned with the x-axis. The radius must be positive. The transformations make it convenient to start with this cylinder and then translate and rotate it to the required position. Translations are applied to the point which started at the origin. For the syntax of the transformations, see the sections on replication and transformation of parts. When projecting a point onto this surface, be sure that it is initialized somewhere away from the axis of symmetry in order to make it clear which direction to project. See **Figure 131**. This is an infinite surface. The graphics will only show a portion of the surface. That portion shown in the graphics changes as the objects in the picture change.

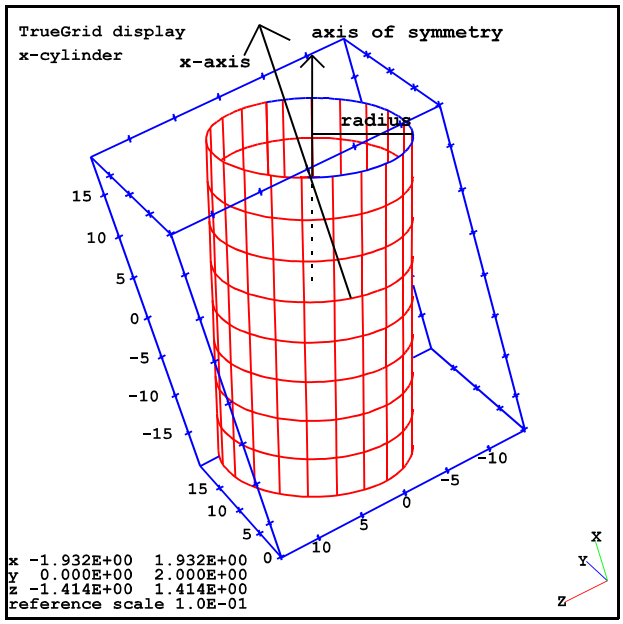


Figure 131 An x-cylinder translated and rotated

Example

```
sd 1 xcy 1 my 1 ry 15 ;
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

xyplan transform an infinite xy-plane

xyplan *trans* ;

where *trans* is a sequence of simple operators to transform the plane.

Remarks

This is the infinite plane defined by $z = 0$. The transform makes it convenient to start with this plane and then translate or rotate it to the desired position. For the syntax of the transformations, see the sections on replication and transformation of parts. The following example produced the picture below. Translations are applied to the plane at the point which originated at *(0,0,0)*.

This is an infinite surface. The graphics will only show a portion of the surface. That portion shown in the graphics changes as the objects in the picture change.

Example

```
sd 1 xyplan rx -15;
```

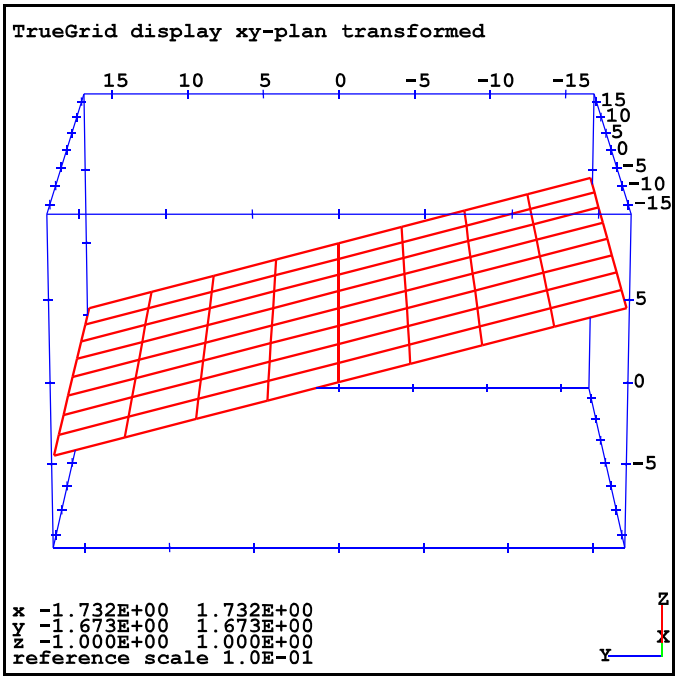


Figure 132 xy-plane rotated

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

ycy to transform an infinite y-axis cylinder

ycy *radius trans* ;

where

radius	radius of the cylinder
trans	sequence of simple operators to transform the coordinates

Remarks

An infinite cylinder starts out with an axis the same as the y-axis. The radius must be positive. The transformations make it convenient to start with this cylinder and then translate and rotate it to the required position. Translations are applied to the point which started at the origin. For the syntax of the transformations, see the sections on replication and transformation of parts. When projecting a point onto this surface, be sure that it is initialized somewhere away from the axis of symmetry to make it clear which direction to project. The following example produced the cylinder below. See **Figure 133**. This is an infinite surface. The graphics will only show a portion of the surface. That portion shown in the graphics changes as the objects in the picture change.

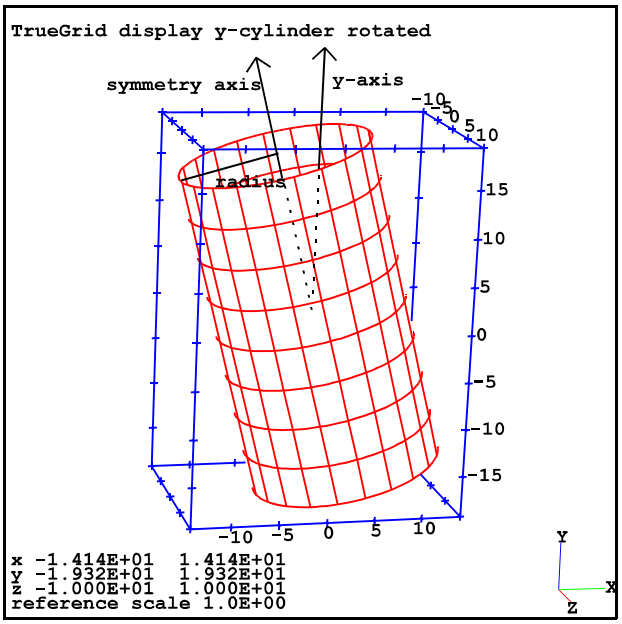


Figure 133 y-cylinder rotated about the z axis

Example

```
sd 1 ycy 10 rz 15;
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

yzplan to transform an infinite yz-plane

yzplan *trans* ;

where *trans* is a sequence of simple operators to transform the plane.

Remarks

This is the infinite plane defined by $x = 0$. The transform makes it convenient to start with this plane and then translate or rotate it to the desired position. For the syntax of the transformations, see the sections on replication and transformation of parts. The following example produced the picture below. Translations are applied to the plane at the point which originated at *(0,0,0)*.

This is an infinite surface. The graphics will only show a portion of the surface. That portion shown in the graphics changes as the objects in the picture change.

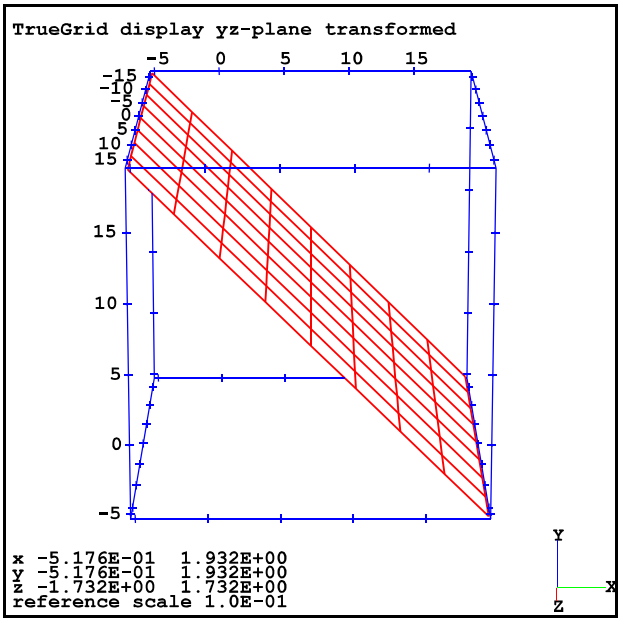


Figure 134 yz-plane moved and rotated

Example

```
sd 1      yzplan mx 1 rz 45;
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

zcy to transform an infinite z-axis cylinder

zcy *radius trans* ;

where

radius	radius of the cylinder
trans	sequence of simple operators to transform the coordinates

Remarks

An infinite cylinder starts out with an axis the same as the z-axis. The radius must be positive. The transformations make it convenient to start with this cylinder and then translate and rotate it to the required position. Translations are applied to the point which started at the origin. For the syntax of the transformations, see the sections on replication and transformation of parts. When projecting a point onto this surface, be sure that it is initialized somewhere away from the axis of symmetry to make it clear which direction to project. The following example produced the picture below. See **Figure 135**. This is an infinite surface. The graphics will only show a portion of the surface. That portion shown in the graphics changes as the objects in the picture change.

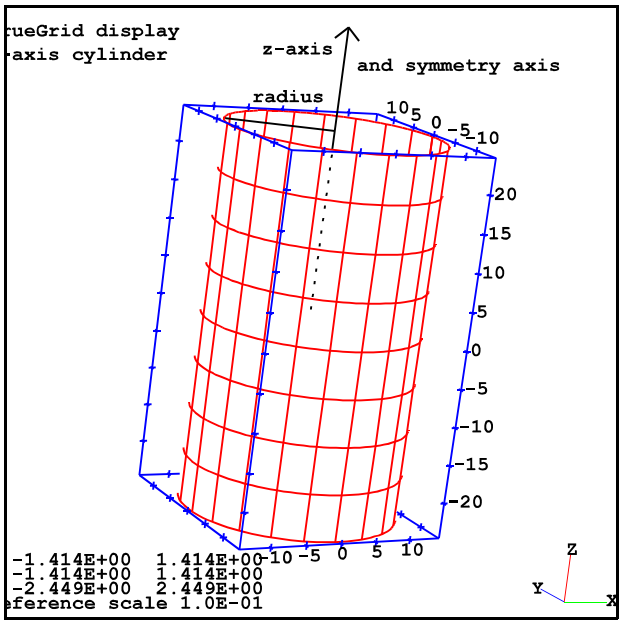


Figure 135 z-cylinder on the z-axis

Example

```
sd 1 zcy [sqrt(2)];
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

zxplan to transform an infinite zx-plane

zxplan *trans* ;

where *trans* is a sequence of simple operators to transform the plane.

Remarks

This is the infinite plane defined by $y = 0$. The transform makes it convenient to start with this plane and then translate or rotate it to the desired position. For the syntax of the transformations, see the sections on replication and transformation of parts. The following example produced the picture below. Translations are applied to the plane at the point which originates at *(0,0,0)*.

This is an infinite surface. The graphics will only show a portion of the surface. That portion shown in the graphics changes as the objects in the picture change.

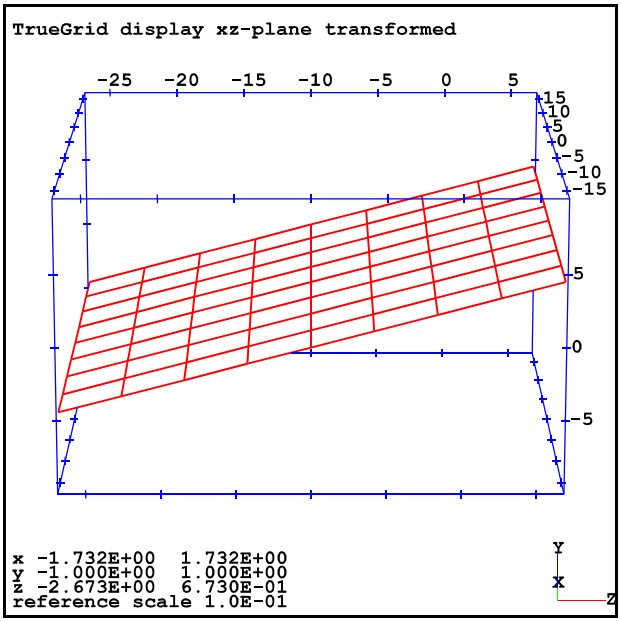


Figure 136 xz-plane rotated and moved

Example

```
sd 1 xzplan rx -15 mz -1;
```

10. Surface Display Commands

These commands let you control which surfaces are displayed in the picture.

When you issue a command to display a surface, you will not necessarily see it. This is because it may lie off-screen. If you issue a surface display command and cannot see the surface, try moving the picture around or zooming in and out. When all else fails, issue a **restore** command (hit the ***Rest*** button).

Many of these commands can be executed using the GUI. In the Environment Window, select the ***Display List*** button. Then choose the ***Surface*** button under the **Apply Action To:** label. The **Action:** buttons will then enable you to reproduce many of the following commands. Some of the display commands in this section are essential. For example, the **asd** command has not been replaced by a button in the GUI. The display commands for many objects, such as 3D curves, parts, and materials,

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

are similar.

Table I

DISPLAY COMMANDS

Menu	Surface	3D Curve	Parts	Material	Interface	Cad	Cad
Type	**Surfaces (sd)**	**3Dcurves (cd)**	**Parts (p)**	**Material (m)**	**Boundary (bb)**	**Groups (grp)**	**Levels (lv)**
Display 1 (d *)	dsd #	dcd #	dp #	dm #	dbb #	dgrp #	dlv #
Add 1 (a *)	asd #	acd #	ap #	am #	abb #	agrp #	alv #
Remove 1 (r *)	rsd #	rcd #	rp #	rm #	rbb #	rgrp #	rlv #
Display Many (d *s)	dsds list;	dcds list ;	dps list ;	dms list ;	dbbs list ;	dgrps list;	dlvs list ;
Add Many (a *s)	asds list ;	acds list ;	aps list ;	ams list ;	abbs list ;	---	---
Remove Many (r *s)	rsds list ;	rcds list ;	rps list ;	rms list ;	rbbs list ;	---	---
Display All (da *)	dasd	dacd	dap	dam	dabb	---	---
Remove All (ra *)	rasd	racd	rap	ram	rabb	---	---

A sequence in a list can be abbreviated by inserting a colon between the first and last numbers. Surfaces can be named. If a surface has been named, then the name can be used instead of its number.

asd add a surface to the picture

asd *surface_#*

Remarks

Surface *surface_#* is added to the display.

asds add surfaces to the picture

asds *surface_numbers* ;

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

The surfaces in the list are added to the displayed.

ansd add neighboring surfaces to the picture

ansd *d θ repeats*

where

d	distance tolerance
θ	tangent plane angle tolerance
repeats	number of times to repeat the process of adding surfaces

Remarks

The purpose of this command is to display the surfaces near to the ones already displayed.
All surfaces are searched other than infinite surfaces without edges, such as a cylinder or paraboloid.

All edges of a surface of a candidate surface are checked for points within a distance *d* of an existing displayed surface. For such nearby points it compare's the candidate surface's tangent plane with the existing surface's tangent plane (at the closest point). If the two tangent planes intersect at an angle less than *θ*, then the two surfaces are near each other.

This is repeated so that the next iteration uses the surfaces just added to the display list to find the next set of near surfaces. If any iteration fails to add any surfaces to the display list, the process terminates.

This command is useful in creating composite surfaces. This is the first step in a four step process.

1. select a set of seed surfaces for display
2. use the **ansd** command to find all neighboring surfaces which are nearly tangent
3. use the **lasd** to list the surface numbers
4. cut and paste this list of surfaces into the **sds** option of the **sd** command

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Examples

This figure shows how the angles between surfaces are measured. The normals to surfaces 1 and 4 differ by 9 degrees. Surface 7 is rejected because the angle is too great.

```
ansd 1 10 1
```

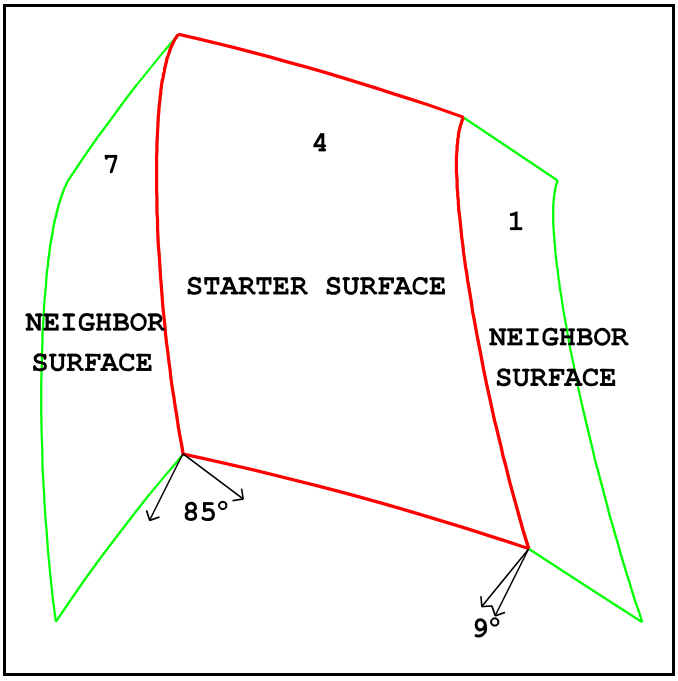


Figure 137 Surface 1 selected adjacent to 4

If you provide a repeat count greater than one, then the search will be repeated. The picture will grow outwards each iteration from the original group of surfaces.

In this example, only surface 1080 was displayed. Then the command

```
ansd .1 3 100
```

was issued to select this set of surfaces.

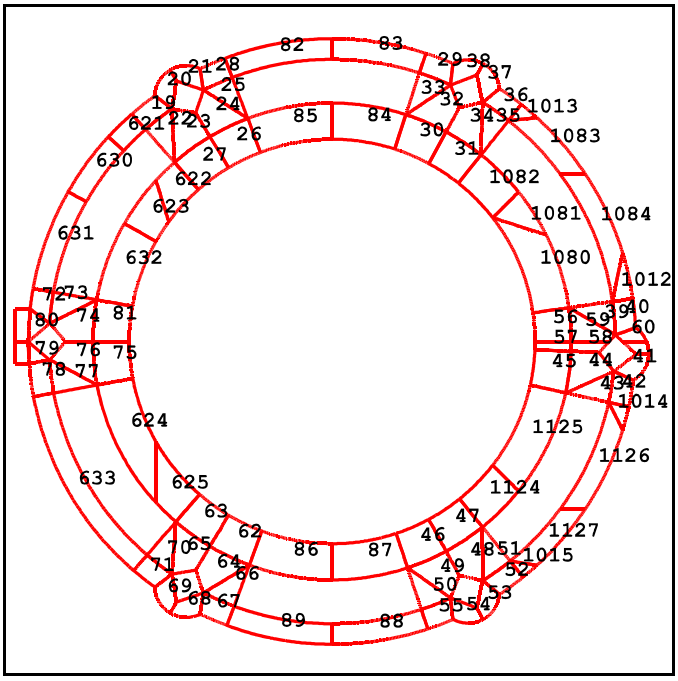


Figure 138 Surface 1080 seeded ANSD

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

dasd display all surfaces in the picture

dasd *(no arguments)*

dsd display a surface in the picture

dsd *surface_#*

Remarks

Displays only one surface – removing all others, if necessary.

dsds display several surfaces in the picture

dsds s_1 s_2 ... s_n ;
where s_i is a numbered surface defined by an **sd** command

Remarks

Displays only the surfaces listed.

lasd list the surfaces in the picture

lasd *(no arguments)*

Remarks

This command is useful in creating composite surfaces. This is the second step in a three step process.

1. select a set of seed surfaces for display
2. use the **lasd** to list the surface numbers
3. cut and paste this list of surfaces into the sds option of the **sd** command

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

rasd remove all surfaces from the picture

rasd *(no arguments)*

rsd remove a surface from the picture

rsd *surface_#*

11. Importing Geometry from CAD Programs

IGES is a standard format for CAD geometry and can be used to pass geometry from a CAD program to **True*Grid*®**. It is advantageous to use IGES geometry because most CAD systems are better than **True*Grid*®** at creating complex geometry. Also, you may already have an IGES model and there is no need to reproduce the effort in creating the geometry in **True*Grid*®**. IGES geometry has the disadvantage that it may be overly complex and reducing the geometry to that which is needed for meshing is more difficult than building the geometry within **True*Grid*®**. This point is mitigated since you have the option, in **True*Grid*®**, to ignore certain features by not projecting a portion of the mesh to certain surfaces. Also, you may not have easy access to a CAD model or you may wish not to learn how to use an available CAD system. Also significant to experienced **True*Grid*®** users is that CAD geometry is not easily made parametric while every aspect of geometry built in **True*Grid*®** can be parametric.

The projection method applied to IGES geometry is superior to other mesh generation techniques in two significant ways. IGES geometry is usually broken into many small components that do not meet perfectly. These small gaps between the geometric components are ignored by the projection method. Also, many meshing techniques may fail or do poorly on some geometry, requiring that the CAD system be reused to break the geometry into smaller pieces with the hope that the smaller pieces will be easily meshed. If successful, these other meshing techniques then must merge the individual parts together. In contrast, the projection method encourages building composite surfaces, done very simply using the **sds** option of the **sd** command for surfaces and the **coedge** command for edges of surfaces, to simplify the geometry and the block structure of the mesh. The mesh generation methods used in **True*Grid*®** have never been shown to fail on any geometry.

There are numerous options when creating an IGES file in a CAD system since there is more than one way to define geometry. In our experience, most CAD systems default to creating an IGES file consisting of solids. This is not acceptable to **True*Grid*®**. **True*Grid*®**'s methods were developed for trimmed surfaces and curves, not solids. Also, in our experience, most CAD operators who create the IGES files are not aware of the options for writing an IGES file. In most cases you need to insist that the IGES file be produced using trimmed surfaces, not solids.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

You can import this IGES geometry created with a CAD program using the **iges** command. This interface is a little different from the interfaces of most CAD systems in that the **True*Grid*®** internal data base is complex and may require significant computations. For this reason, you can save this data base, once generated, in a binary IGES file. In the first instance, you use the **iges** command to import the geometry followed by the **saveiges** command to save the binary IGES data base. Subsequently, you use the **useiges** command to import the binary IGES data base prior to using the same **iges** command that you used in the first instance. If you take these extra steps, you can save yourself considerable time.

First usage:

```
iges file.igs 1 1;
saveiges file.bin
```

All subsequent usage:

```
useiges file.bin
iges file.igs 1 1;
```

Although IGES is an international standard, there is no enforcement of the standard and no quality assurance. CAD systems, on occasions, produce faulty IGES files because of bugs in the CAD system, deviations from the IGES standard, and mathematical abuses. Our policy is to develop algorithms that work around these problems produced by the CAD systems. Since the CAD systems are constantly evolving, **True*Grid*®** must be constantly evolving as well. So we ask that if you find a problem when importing an IGES file, please contact XYZ Scientific applications, Inc.. The best way to help is to send the IGES file. This may be difficult because the IGES file may be too large or it may be proprietary or classified. Alternatively, you can use the WINDOWS® IGES Inspection Utility (**igesfind.exe**) or the UNIX®/LINUX® **igesfind** program supplied by XYZ Scientific Applications, Inc. to extract a specific surface from an IGES file. Then send just that one surface to XYZ Scientific Applications, Inc. by email to info@truegrid.com. For example, suppose that surface 123 of an IGES file called widget.igs causes a problem in **True*Grid*®**. With the IGES Inspection Utility on WINDOWS®, select that numbered surface and and click on *save as* to create a new IGES file with just the surface numbered 123. On a UNIX®/LINUX® system type

igesfind widget.igs -surface 123 > s123.igs

to produce an IGES file s123.igs with only the surface numbered 123.

Geometric entities in an IGES file can depend on other entities, forming a hierarchy. When the geometric entities are numbered, room is left in the sequence for those entities which form the basis for others. If you use the **iges** command, then entities which form the basis for other entities are not

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

rendered. For example, the curves used to trim a surface are not evaluated since the entity will be embedded in the trimmed surface as one of its edges. You can use one of the other IGES evaluation commands discussed below if you wish to use these basis entities. There are two important entities which fall in this category. They are composite curves (entity type number 102) and trimmed surfaces (entity type number 144). You will notice, when importing an IGES file using the **iges** command, that the underlying composite and component curves as well as the untrimmed surface are all assigned numbers, but they are not rendered. There is no loss of data, since all of the information about these component entities are built into the trimmed version of the surface.

The following is a list of the supported IGES entity types. The bounded surface entity (143) and its associated bounded entity (type 141), which are used to form solids, are not supported.

3D curves	line	entity type 108
	circular arc	entity type 100
	conic arc	entity type 104
	parametric spline	entity type 112
	NURBS curve	entity type 126
	composite	entity type 102
	copious data	entity type 106
Surfaces	plane including bounded plane	entity type 108
	ruled surface	entity type 118
	surface of revolution	entity type 120
	tabulated cylinder	entity type 122
	parametric spline	entity type 114
	NURBS surface	entity type 128
	trimmed surface	entity types 144 & 142
	offset surface	entity type 140
Others	transformation	entity type 124
	level	N/A
	associativity	entity type 302
	subfigure	entity type 308

NURBS (Non-Uniform Rational B-Spline) surfaces are a type of surface that has become, in most CAD systems, the preferred type of surface because of their versatility. Planar surfaces can also be found in an IGES file. There are other types of surfaces such as ruled, parametric spline, revolved, and tabulated. These three basic types of surfaces, NURBS, planes, and others, are each assigned an IGES sequence number, according to their type, so that they can be referenced for rendering purposes. These IGES sequence numbers are different from the number you assign a surface. For example, the first NURBS IGES surface may be assigned to be surface number 2 in **True*Grid*®**

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

because you may already have a surface number 1. The untrimmed NURBS surfaces are assigned sequence numbers first, followed by the trimmed versions of these surfaces. As is noted above, the untrimmed version of a NURBS surface will not be automatically rendered if there is also a trimmed version of that surface. In the same manner, planes and other surface types are sequenced and rendered. Then IGES curves are numbered and rendered in the order that they are found in the IGES file. Although an IGES curve may be 2D or 3D, they are all treated as 3D curves, with 2D curves being assigned a third constant coordinate. Since most 2D curves are usually used to define a trimmed surface, they are not automatically rendered.

When there are multiple IGES files being used, the IGES sequence numbering of NURBS, planes, other surfaces, and curves start where they left off on from the previous IGES surface.

You may have difficulty reading an IGES file created on a different machine. This is because WINTEL systems use a different end-of-line and carriage return from UNIX/LINUX systems. One way to solve this problem is to use a text editor to remove the extra character at the end of the line and rewrite it. There may also be utilities on your system to do this conversion. If you use ftp to transfer the file, be sure to leave it in ASCII or character mode (not binary).

You can display rendered IGES curves with commands such as **dcd**, **dacd**, **acd**, **acds**, **rcd**, **rdds**, and **dcds**. You can display rendered IGES surfaces with commands such as **dsd**, **dasd**, **asd**, **asds**, **rsd**, **rsds**, **dsds**. You can place the edge of the mesh along a rendered IGES curve with the **curs** command. You can place a face of the mesh on a rendered IGES surface with the **sf** command.

The default accuracy in rendering the geometry is approximately 3.5 digits of accuracy. This rendering tolerance is relative to the size of the geometric entity being rendered. The accuracy of the rendering affects the time it takes to do the rendering and the amount of memory needed to store the rendering data. This also affects the size of an IGES binary file written using the **saveiges** command. It also affects the accuracy of the default projection to surfaces. Both the accuracy of the rendering and the accuracy of the projection can be controlled, independently, using the **getol** and **accuracy** commands, respectively. In particular, you may wish to decrease the accuracy of the rendering to speed the rendering and decrease the memory requirements while increasing the projection accuracy. This is a classic example of the tradeoff of memory and computations.

You can also read in geometry which has been stored in a simple polygon data format using the **vpsd** command or the **stl** or **bstl** options of the **sd** command. These types of surfaces are not affected by the **getol** command.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

iges render geometry in an IGES file

iges *file surface_# curve_# transformations ;*

where

file IGES file name including the path relative to the home directory

surface_# surface number to be assigned to the first surface in the file

curve_# 3D curve number to be assigned to the first 3D curve in the file

transformations list of coordinate transformations to be applied to all geometry

where a transform can be

mx *x_offset* for translation in the x direction

my *y_offset* for translation in the y direction

mz *z_offset* for translation in the z direction

v *x_offset y_offset z_offset* for a general translation

rx θ for rotation about the x axis

ry θ for rotation about the y axis

rz θ for rotation about the z axis

raxis *angle x0 y0 z0 xn yn zn* axis of rotation

rxy for reflection about the x-y plane

ryx for reflection about the y-x plane

rzx for reflection about the z-x plane

tf *origin x-axis y-axis*

where each of the arguments consist of a coordinate type followed by coordinate information:

rt *x y z* Cartesian coordinates

cy *rho theta z* cylindrical coordinates

sp *rho theta phi* spherical coordinates

pt *c.i* label of a labeled point from a 3D curve

pt *s.i.j* label of a labeled point from a surface

ftf *1st_origin 1st_x-axis 1st_y-axis 2nd_origin 2nd_x-axis 2nd_y-axis*

where each of the arguments consist of a coordinate type followed by coordinate information:

rt *x y z* Cartesian coordinates

cy *rho theta z* cylindrical coordinates

sp *rho theta phi* spherical coordinates

pt *c.i* label of a labeled point from a 3D curve

pt *s.i.j* label of a labeled point from a surface

inv invert the transformation preceding this option

csca *scale_factor* to scale all coordinates

xsca *scale_factor* to scale the x coordinate

ysca *scale_factor* to scale the y coordinate

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

zsca *scale_factor* to scale the z coordinate

Remarks

It is inefficient to start with a high number for the curves or surfaces.

The number of NURBS, planes, other surface, and curves found in an IGES file is reported in the text window and the tsave file.

This command is the simplest way to import IGES geometry. You may use the **igesfile** to be selective on which entities to render. Although this may be faster, because you only render the geometry you want, you must have already decided what curves and surfaces you want. This can only be decided by using the **iges** command first so that you can see the surfaces and curves to make your choices.

To save the rendering calculations for future re-use, use the **saveiges** command. The next time you want the surfaces and 3D curves from this IGES file, issue the **useiges** command before issuing the **iges** or other IGES rendering command.

In the WINDOWS operating system, when you click on the iges command in the menus, you will get a browser that will aid you in selecting the IGES file.

igesfile open an IGES file

igesfile *filename*

Remarks

When you enter an **igesfile** command, the IGES file is opened. The geometry in the IGES file is reported, but not rendered. To render curves, use the **curd** command using the **igc** option, or with the **igescd** command. To render a NURBS surface, use the **sd** command using the **nurbs** option, or with the **nurbsd** command. For a plane, evaluate it with the **sd** command using the **igesp** option, or with the **igespd** command. For any other type of IGES surface, evaluate it with the **sd** command using the **igess** option, or with the **igessd** command.

This command should only be used by an experienced user. It requires prior knowledge of the geometry in the specified file. This command has the advantage, over the iges command, of selecting out a subset of the geometric entities for rendering. It is advantageous only when the subset is relatively small.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

The number of NURBS, planes, other surface, and curves found in an IGES file is reported in the text window and the tsave file.

igescd render a sequence of IGES curves

igescd *curve$_1$ curve$_n$ first_curve transformations ;*

where

curve$_1$ is the sequence number in the IGES file of the first curve to be evaluated

curve$_n$ is the sequence number in the IGES file of the last curve to be evaluated

first_curve is the **True*Grid*®** curve number to be assigned to the first IGES curve

transformations is a list of any of the following coordinate transformations:

mx *x* for x translation
my *y* for y translation
mz *z* for z translation
v *x y z* for a general translation
rx *θ* for rotation about the x-axis
ry *θ* for rotation about the y-axis
rz *θ* for rotation about the z-axis
raxis *angle x0 y0 z0 xn yn zn* axis of rotation
rxy for reflection about the x-y plane
ryz for reflection about the y-z plane
rzx for reflection about the z-x plane
tf *origin x-axis y-axis*

where each of the arguments consist of a coordinate type followed by coordinate information:

rt *x y z*	Cartesian coordinates
cy *rho theta z*	cylindrical coordinates
sp *rho theta phi*	spherical coordinates
pt *c.i*	label of a labeled point from a 3D curve
pt *s.i.j*	label of a labeled point from a surface

ftf *1st_origin 1st_x-axis 1st_y-axis 2nd_origin 2nd_x-axis 2nd_y-axis*

where each of the arguments consist of a coordinate type followed by coordinate information:

rt *x y z*	Cartesian coordinates
cy *rho theta z*	cylindrical coordinates
sp *rho theta phi*	spherical coordinates
pt *c.i*	label of a labeled point from a 3D curve
pt *s.i.j*	label of a labeled point from a surface

inv invert the present transformation

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

csca *scale_factor*	to scale all coordinates
xsca *scale_factor*	to scale the x coordinate
ysca *scale_factor*	to scale the y coordinate
zsca *scale_factor*	to scale the z coordinate

Remarks

First specify the IGES file with the **igesfile** or the **iges** command.

It is inefficient to start with a high number for the curves.

A unique **True*Grid*®** 3D curve number will be assign to each rendered IGES curve, starting with the specified number. All specified curves will be rendered, even those used in the construction or rendering of more complex geometric entities.

igeslbls use IGES surface labels to name surfaces

igeslbls *flag*

where *flag* can be **on** or **off**

Remarks

The default is **off**.

Sometimes, an IGES file includes labels for the surfaces. When several surfaces are found to have the same label, they are treated as a composite surface. The result would be the same if the **sd** command were issued using the **sds** option to combine the several surfaces into one. Surface numbers are assigned for internal bookkeeping. It will grab the next available surface number to do this. You will be notified for each composite surface. Names must have some alphabetic characters in them and no spaces. Other special characters like : and ; should also be avoided, although the implications of using none alphanumeric characters has not been explored. This feature is case insensitive. Only the first 8 characters of a name are used.

One can add surfaces to a composite by simply naming it in the **sd** command when the surface is defined.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

igespd render a sequence of IGES planes

igespd *plane$_1$ plane$_n$ first_surface transformations*

where

plane$_1$ sequence number in the IGES file of the first plane to be evaluated

plane$_n$ sequence number of the last plane to be evaluated

first_surface **True*Grid*®** surface number to be assigned to the first plane

transformations is a list of any of the following coordinate transformations:

mx *x* for x translation

my *y* for y translation

mz *z* for z translation

v *x y z* for a general translation

rx θ for rotation about the x-axis

ry θ for rotation about the y-axis

rz θ for rotation about the z-axis

raxis *angle x0 y0 z0 xn yn zn* axis of rotation

rxy for reflection about the x-y plane

ryz for reflection about the y-z plane

rzx for reflection about the z-x plane

tf *origin x-axis y-axis*

where each of the arguments consist of a coordinate type followed by coordinate information:

rt *x y z*	Cartesian coordinates
cy *rho theta z*	cylindrical coordinates
sp *rho theta phi*	spherical coordinates
pt *c.i*	label of a labeled point from a 3D curve
pt *s.i.j*	label of a labeled point from a surface

ftf *1st_origin 1st_x-axis 1st_y-axis 2nd_origin 2nd_x-axis 2nd_y-axis*

where each of the arguments consist of a coordinate type followed by coordinate information:

rt *x y z*	Cartesian coordinates
cy *rho theta z*	cylindrical coordinates
sp *rho theta phi*	spherical coordinates
pt *c.i*	label of a labeled point from a 3D curve
pt *s.i.j*	label of a labeled point from a surface

inv invert the present transformation

csca *scale_factor* to scale all coordinates

xsca *scale_factor* to scale the x coordinate

ysca *scale_factor* to scale the y coordinate

zsca *scale_factor* to scale the z coordinate

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

First specify the IGES file with the **igesfile** or **iges** command.

It is inefficient to start with a high number for the surfaces.

A unique **True*Grid*®** surface number will be assign to each rendered IGES plane, beginning with the specified number.Everyl requested plane will be rendered, even if it is untrimmed and forms the basis for a trimmed surface.

igessd render a sequence of IGES surfaces

igessd *surface$_1$ surface$_n$ first_surface transformations*

where

surface$_1$ is the first IGES surface number to be selected

surface$_n$ is the last IGES surface number to be selected

first_surface is the first **True*Grid*®** surface definition number

transformations is a list of any of the following coordinate transformations:

mx *x* for x translation
my *y* for y translation
mz *z* for z translation
v *x y z* for a general translation
rx θ for rotation about the x-axis
ry θ for rotation about the y-axis
rz θ for rotation about the z-axis
raxis *angle x0 y0 z0 xn yn zn* axis of rotation
rxy for reflection about the x-y plane
ryz for reflection about the y-z plane
rzx for reflection about the z-x plane
tf *origin x-axis y-axis*

where each of the arguments consist of a coordinate type followed by coordinate information:

rt *x y z*	Cartesian coordinates
cy *rho theta z*	cylindrical coordinates
sp *rho theta phi*	spherical coordinates
pt *c.i*	label of a labeled point from a 3D curve
pt *s.i.j*	label of a labeled point from a surface

ftf *1st_origin 1st_x-axis 1st_y-axis 2nd_origin 2nd_x-axis 2nd_y-axis*

where each of the arguments consist of a coordinate type followed by

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

coordinate information:

rt *x y z*	Cartesian coordinates
cy *rho theta z*	cylindrical coordinates
sp *rho theta phi*	spherical coordinates
pt *c.i*	label of a labeled point from a 3D curve
pt *s.i.j*	label of a labeled point from a surface

inv	invert the present transformation
csca *scale_factor*	to scale all coordinates
xsca *scale_factor*	to scale the x coordinate
ysca *scale_factor*	to scale the y coordinate
zsca *scale_factor*	to scale the z coordinate

Remarks

First specify the IGES file with the **igesfile** or **iges** command.

It is inefficient to start with a high number for the surfaces..

A unique **True*Grid*®** surface number will be assign to each rendered IGES surface, beginning with the specified number. Each specified surface will be rendered, even if it is untrimmed and forms the basis for a trimmed surface.

To evaluate NURBS surfaces, use the **nurbsd** command. In order to evaluate planes, use the **igespd** command. This command is for all other IGES surfaces that **True*Grid*®** supports.

nurbsd render a sequence of IGES NURBS surfaces

nurbsd $surface_1$ $surface_n$ *first_surface transformations ;*

where

$surface_1$	is the IGES file sequence number of the first NURBS surface used
$surface_n$	is the IGES file sequence number of the last NURBS surface used
first_surface	is the first **True*Grid*®** surface definition number
transformations	is an optional list of transformations to be applied to the surfaces:

mx *x* for x translation
my *y* for y translation
mz *z* for z translation
v *x y z* for a general translation
rx θ for rotation about the x-axis
ry θ for rotation about the y-axis

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

rz θ for rotation about the z-axis
raxis *angle x0 y0 z0 xn yn zn* axis of rotation
rxy for reflection about the x-y plane
ryz for reflection about the y-z plane
rzx for reflection about the z-x plane
tf *origin x-axis y-axis*
where each of the arguments consist of a coordinate type followed by coordinate information:

rt *x y z*	Cartesian coordinates
cy *rho theta z*	cylindrical coordinates
sp *rho theta phi*	spherical coordinates
pt *c.i*	label of a labeled point from a 3D curve
pt *s.i.j*	label of a labeled point from a surface

ftf *1st_origin 1st_x-axis 1st_y-axis 2nd_origin 2nd_x-axis 2nd_y-axis*
where each of the arguments consist of a coordinate type followed by coordinate information:

rt *x y z*	Cartesian coordinates
cy *rho theta z*	cylindrical coordinates
sp *rho theta phi*	spherical coordinates
pt *c.i*	label of a labeled point from a 3D curve
pt *s.i.j*	label of a labeled point from a surface

inv invert the present transformation
csca *scale_factor* to scale all coordinates
xsca *scale_factor* to scale the x coordinate
ysca *scale_factor* to scale the y coordinate
zsca *scale_factor* to scale the z coordinate

Remarks

This command applies to the IGES file you selected with the **igesfile** or **iges** command.

It is inefficient to start with a high number for the surfaces.

A unique **True*Grid*®** surface will be assigned to each rendered IGES NURBS surface, starting with the specified number. Each specified surface will be rendered, even if it is untrimmed and forms the basis for a trimmed surface.

In order to evaluate planes, use the **igespd** command. Use the **igessd** command to render other IGES surface types.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

saveiges save IGES binary data

saveiges *file*

where *file* is the file name to contain the binary IGES surface and 3D curve data

Remarks

If your IGES file has a large number of surfaces and 3D curves that you need to reuse, it may be time-consuming for **True*Grid*®** to render them each time you rerun your batch file. With this command you can save the evaluations so that in the future you can quickly access the surfaces and curves. This command is used after the IGES geometry has been rendered. All IGES geometric entities (surfaces and 3D curves) that have been rendered will be stored in the binary data file. Each surface or 3D curve from an IGES file is assigned a key in this binary data file. When this binary data is retrieved using the **useiges** prior to either the **iges** or **igesfile** command, the surfaces and curves in the binary data file will be matched to the corresponding geometric entities from the appropriate IGES files. The name of the IGES file is used to form this key for each surface and curve and it is required that the IGES file name not be changed so that the correct correspondence between IGES entities and the binary data can be established.

trimming controls the trimmed surface algorithm

trimming *option*

where the *option* can be

on	this activates the trimming
off	this deactivates the trimming

Remarks

This feature is not usually needed. It is used for backward compatibility and for debugging. The default is **on**.

ltrim set the surface trimming work space

ltrim *size*

where *size* is the number of words to use in the trimming algorithm

Remarks

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

This command is not usually needed. The default is 2000000 words. Only use this command to increase the work space if the code issues a warning that it had difficulties trimming a surface and perhaps increasing the work space may help.

```
warning - not enough work space to trim surface.
          try increasing the work space from 2000000
          by using the ltrim command.
```

This can occur if there are some small details in the trimming curves or if there are many interior trimming curves. **True*Grid*®** tries to trim the surface by first using the domain curves. Sometimes these trimming curves are not supplied in the IGES file. If they are not supplied or if they fail, then **True*Grid*®** attempts to trim the surface with the model space curves.

You can also try reducing the resolution of the trimming using the **geto**l command.

Often, when this problem occurs, the IGES file includes a poorly defined trimmed surface. For instance, the surface component trimming curves do not meet at their respective end points or the trimming curve has a loop causing the curve to intersect itself. These problems violate the IGES standard and sometimes occur in IGES files. Most of these problems with trimming curves are automatically fixed by **True*Grid*®** and you are never warned about it.

The best way to determine if the trimming curves are bad is to use the **igescd** to render the trimming curves. Then inspect the curves for these flaws. If you have a large IGES file, it will be easier if you first extract the offending surface using the IGESFIND utility that is supplied with each **True*Grid*®** distribution.

If the trimming curves are bad, perhaps you can modify the IGES file using the CAD system which created the trimmed surfaces originally. If it appears that there is no problem with the trimming curves, and you have tried increasing the size of the work space with no results, please contact XYZ.

useiges use saved binary IGES data

useiges *file*

where *file* is the name of the binary IGES file made by the **saveiges** command.

Remarks

If your IGES file has a large number of surfaces and 3D curves that you need to use, it may be time-consuming for **True*Grid*®** to evaluate them all. You can save the evaluations with **saveiges** and then quickly read them back in with this command.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

This command is used before the IGES file is opened using either **iges** or **igesfile**. During a previous run with **True*Grid*®**, any IGES geometric entities (surfaces and curves) that were rendered (were visible) and were stored in the binary data file will be retrieved. Each surface or curve from an IGES file was assigned a key in this binary data file. When this binary data is retrieved using the **useiges** followed by either the **iges** or **igesfile** command, the surfaces and curves in the binary data file are matched to the corresponding geometric entities from the appropriate IGES files. The name of the IGES file is used to form this key for each surface and curve and it is required that the IGES file name not be changed so that the correct correspondence between the IGES entities and the binary data can be established. Up to 100 binary files can be read in with this command.

vpsd import ViewPoint surfaces

vpsd *first_surface nodal_file connectivity_file transformations ;*

where

first_surface is the **True*Grid*®** surface number to be assigned to the first surface read from the file,

nodal_file is the ViewPoint nodal file,

connectivity_file is the ViewPoint connectivity file, and

transformations is a list of coordinate transformations to be applied to all the curves and surfaces. Choose them from among:

- **mx** *x_offset* for translation in the x direction
- **my** *y_offset* for translation in the y direction
- **mz** *z_offset* for translation in the z direction
- **v** *x_offset y_offset z_offset* for a general translation along a vector
- **rx** θ for rotation about the x axis
- **ry** θ for rotation about the y axis
- **rz** θ for rotation about the z axis
- **raxis** *angle x0 y0 z0 xn yn zn* axis of rotation
- **rxy** for reflection about the x-y plane
- **ryx** for reflection about the y-x plane
- **rzx** for reflection about the z-x plane
- **tf** *origin x-axis y-axis*

  where each of the arguments consist of a coordinate type followed by coordinate information:

  - **rt** *x y z* Cartesian coordinates
  - **cy** *rho theta z* cylindrical coordinates
  - **sp** *rho theta phi* spherical coordinates
  - **pt** *c.i* label of a labeled point from a 3D curve
  - **pt** *s.i.j* label of a labeled point from a surface

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

ftf *1st_origin 1st_x-axis 1st_y-axis 2nd_origin 2nd_x-axis 2nd_y-axis*
where each of the arguments consist of a coordinate type and coordinate data:

rt *x y z*	Cartesian coordinates
cy *rho theta z*	cylindrical coordinates
sp *rho theta phi*	spherical coordinates
pt *c.i*	label of a labeled point from a 3D curve
pt *s.i.j*	label of a labeled point from a surface

inv invert the present transformation

csca *scale_factor*	to scale all coordinates
xsca *scale_factor*	to scale the x coordinate
ysca *scale_factor*	to scale the y coordinate
zsca *scale_factor*	to scale the z coordinate

Remarks

If you need to write your own software to generate polygonal surfaces, you can write your output as ViewPoint files. These file formats are a good choice because of their simplicity. The first file contains the point data for the polygons. Each record is a point in three-dimensional space. There are 4 fields, each field separated by a comma. The first field is the node number, followed by the x, y, and z-coordinates. The second file contains the connectivity of the nodes. Each record defines a polygon. The fields are space delimited. The first field is the name of the set containing this polygon. This is followed by the ordered list of node numbers defining the polygon.

The polygons in these surfaces should be almost planar. When they are read in, they are reduced to triangles, unless they are quadrilaterals. The shape of the interior of a many sided polygon will be ambiguous, if the nodes are not co-planar.

Each unique name in the connectivity file produces a surface in **True*Grid*®**. The polygons do not have to be connected. However, the edges of polygons which are not shared by exactly two polygons are treated as boundary edges. These boundary edges are sewn together to form surface edges (see **labels** and **curd**).

Example

The following example creates 2 surfaces, **inner** and **outer**. The nodal and connectivity files are listed below. These files were read with the command

```
vpsd  1  vpsd.nodes vpsd.elem ;
```

See below for a picture of the resulting surfaces.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Nodal file: vpsd.nodes

```
      1,0.87,0.50,3.00
      2,0.00,1.00,3.00
      3,-0.87,0.50,3.00
      4,-0.87,-0.50,3.00
      5,0.00,-1.00,3.00
      6,0.87,-0.50,3.00
      7,1.30,0.75,2.75
      8,0.00,1.50,2.75
      9,-1.30,0.75,2.75
      10,-1.30,-0.75,2.75
      11,0.00,-1.50,2.75
      12,1.30,-0.75,2.75
      13,1.97,-0.35,2.5
      14,1.97,0.35,2.5
      15,1.29,1.53,2.5
      16,0.68,1.88,2.5
      17,-0.68,1.88,2.5
      18,-1.29,1.53,2.5
      19,-1.97,0.35,2.5
      20,-1.97,-0.35,2.5
      21,-1.29,-1.53,2.5
      22,-0.68,-1.88,2.5
      23,0.68,-1.88,2.5
      24,1.29,-1.53,2.5
      25,2.17,1.25,2.25
      26,0.00,2.50,2.25
      27,-2.17,1.25,2.25
      28,-2.17,-1.25,2.25
      29,0.00,-2.50,2.25
      30,2.17,-1.25,2.25
Connectivity file: vpsd.elem

      outer 1 6 12 13 14 7
      outer 1 7 15 16 8 2
      outer 2 8 17 18 9 3
      outer 3 9 19 20 10 4
      outer 4 10 21 22 11 5
      outer 5 11 23 24 12 6
      inner  1 2 3 4 5 6
      outer 7 14 25 15
      outer 8 16 26 17
      outer 9 18 27 19
      outer 10 20 28 21
      outer 11 22 29 23
      outer 12 24 30 13
```

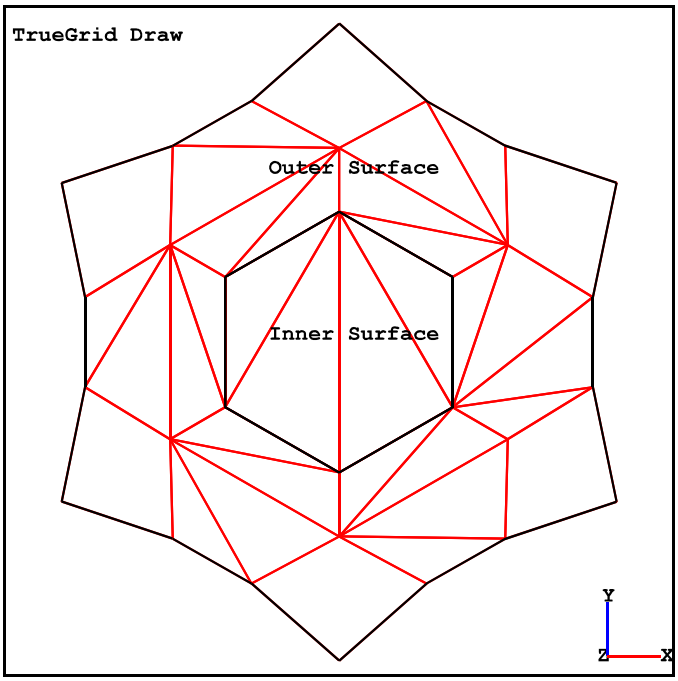


Figure 139 Polygon Surfaces

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

wrsd write surface using the ViewPoint format

wrsd *node_file element_file surface_list* ;

where

node_file	file name to contain the nodal data for the surfaces
element_file	file name to contain the connectivity data for the surfaces
list	list of **True*Grid*®** surface numbers

Remarks

The files produced with this command can be read into **True*Grid*®** using the **vpsd** command.

12. Displaying Geometry from CAD Programs

IGES is the acronym for the Initial Graphics Exchange Specification which is an international standard interface for communicating geometry between Computer Aided Design (CAD) programs. When **True*Grid*®** reads such a file, it writes a short report of the geometry found in the file. Two types of objects that might be found in an IGES file are used to organize the geometry into sets. These objects must be created by the CAD operator before **True*Grid*®** can take advantage of them. The first is a level. Every geometric entity in an IGES file can belong to a level. **True*Grid*®** maintains the levels and you can select levels for graphics. The associativity group can be any set of geometric entities. A geometry entity can belong to many associativity groups. **True*Grid*®** maintains these groups and you can select groups for graphics. The following example of reading an IGES file demonstrates the identification of levels and groups.

```
iges maison.igs 1 1;

   The following IGES levels were found:
      1    2
   IGES file contained    34 NURBS surfaces from     1 to    34
   IGES file contained   192 curves from     1 to   192
   IGES file contained     2 groups from     1 to     2
```

When a sequence of levels or groups are needed, you only need to give the first and last numbers separated by a colon.

When there are many surfaces in an IGES model, the first step in creating a mesh is to organize the geometry. Usually, there are many smaller surfaces that should be grouped to form larger composite surfaces because they touch each other tangentially. These larger surfaces become a more intuitive decomposition of the geometry. These larger composite surfaces also make it easier to design and build the block topology of the mesh because there is no longer the need to match the mesh

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

boundaries to arbitrary component surface boundaries. If the CAD operator can use the associativity groups and levels in anticipation of building the mesh, the time spent using **True*Grid*®** to organize the geometry can be greatly reduced. If a set of surfaces are in the same level or group, to be formed into a composite surface, take the following steps:

1. Display only that level using the **dlv** command or that group using the **dgrp** command.
2. Use the **lasd** command to list all the surfaces in the picture.
3. Use the **sds** option of the **sd** command and cut and paste the list of surfaces into the command.

alv add a CAD level to the picture

alv *level*

where *level* is the number of the CAD level to be added.

Remarks

This adds to the picture all defined surfaces resulting from CAD IGES surfaces and all 3D curves resulting from CAD IGES curves defined within the specified level.

dlv display a single CAD level in the picture

dlv *level*

where *level* is the number of the CAD level to be displayed

Remarks

This will display all defined surfaces resulting from CAD IGES surfaces and all 3D curves resulting from CAD IGES curves defined within the specified level. This command will remove all other surfaces and 3D curves from the picture.

dlvs display several CAD levels in the picture

dlvs $level_1$ $level_2$ ... $level_n$ **;**

where

$level_i$ is the number of one of the CAD levels to be displayed.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

This command will display all defined surfaces resulting from a CAD IGES surface and all 3D curves resulting from a CAD IGES curve defined within any of the specified levels. This will remove all other surfaces and 3D curves from the picture.

rlv remove a CAD level from the picture

rlv *level*
where *level* is the number of the CAD level to be removed.

Remarks

This command will stop displaying the defined surfaces resulting from any CAD IGES surfaces and all 3D curves resulting from any CAD IGES curves defined within the specified level.

agrp add a CAD group to the picture

agrp *group*
where *group* is the number of the CAD group to be added.

Remarks

This command will add to the picture all defined surfaces resulting from CAD IGES surfaces and all 3D curves resulting from CAD IGES curves defined within the specified group.

dgrp display a single CAD group in the picture

dgrp *group*
where
group is the number of the CAD group to be displayed

Remarks

This command will display all defined surfaces resulting from CAD IGES surfaces and all 3D curves resulting from CAD IGES curves defined within the specified group. This command will remove all other surfaces and 3D curves from the picture.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

dgrps display several CAD groups in the picture

dgrps *group*$_1$ *group*$_2$ ... *group*$_n$ **;**
where *group*$_i$ is the number of one of the CAD groups to be displayed.

Remarks

This command will display all defined surfaces resulting from a CAD IGES surface and all 3D curves resulting from a CAD IGES curve defined within any of the specified groups. This command will remove all other surfaces and 3D curves from the picture.

rgrp remove a CAD group from the picture

rgrp *group*
where *group* is the number of the CAD group to be removed.

Remarks

This command will stop displaying the defined surfaces resulting from any CAD IGES surfaces and all 3D curves resulting from any CAD IGES curves defined within the specified group.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

II. Assembly Commands - Merge Phase

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

1. Merging Parts

In the Merge Phase, nodes that are close to one another are merged into a single node. The merging commands allow you to define how close is close. All tolerances are in absolute distances. There are commands for specifying tolerances for the general merging of all nodes over all parts or just nodes on the exterior faces of the mesh. There are commands for specifying the tolerances for the special merging of nodes between parts or within a part. These special tolerances override the general ones. If no tolerance commands are specified, then no merging is done. However, the Merge Phase must be entered in order to build the node map which is used to generate the output.

Invocation of a tolerance command (**t**, **tp**, **st**, **stp**) within the Merge Phase causes an immediate merging of nodes. These commands can also be invoked within any phase; when the Merge Phase is entered, those tolerance commands are immediately executed or re-executed as the case may be. The merge process is always performed on the nodes in their original (prior to any merging) state. Merging is not cumulative. If you leave the Merge Phase and reenter it, all merging is recalculated with what ever new parts that have been added. This lets you interactively experiment with merging and tolerances. Setting a tolerance to a negative value is an easy way to restore the nodes to their original states. Graphical displays of the mesh in the Merge Phase always reflect the results of any merging.

Nodes are merged depending on the distance between them. If a node lies within a tolerance distance of more than one other node, then it is merged with the closest one. When merging several nodes into one node, the first-defined node survives. This can be overridden by the **bptol** command. Nodes within a joint and across the two sides of a sliding surface are not merged. When the first merging of nodes occurs, a sliding interface table is calculated which is used in the merging process. This table is written to the screen and to the save file and is intended as diagnostics. The following is a sample of that table:

SLIDING INTERFACE SUMMARY

Surf	S-node	S-lseg	S-qseg	M-node	M-lseg	M-qseg
1	105	84	0	468	418	0
2	232	0	52	468	418	0
3	221	0	0	390	304	0
4	221	0	0	390	304	0
5	158	0	0	120	88	0
6	158	30	30	120	88	0
7	204	102	0	204	102	0
8	232	0	52	90	52	0
9	101	18	18	161	132	0
10	101	18	18	161	132	0
11	548	120	120	3216	0	1056

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
12      133       84        0      161      132        0
13      133       84        0      161      132        0
14      308      240        0     3216        0     1056
```

This table is organized by the sliding interface number on the right. Columns 2, 3, and 4 are datum pertaining to the slave side on the interface; columns 5, 6, and 7 to the master side. Columns 2 and 5 (`S-node` and `M-node`) are node counts. Columns 3 and 6 (`S-lseg` and `M-lseg`) are linear face counts, and columns 4 and 7 (`S-qseg` and `M-qseg`) are quadratic face counts.

A table of merged nodes is always written after the **tp** or **stp** commands are executed.

```
                MERGED NODES SUMMARY
 12 nodes merged between parts        1 and        2
 16 nodes merged between parts        2 and        2
 12 nodes merged between parts        1 and        3
 16 nodes merged between parts        3 and        3
216 nodes merged between parts        4 and        4
 30 nodes merged between parts        7 and        7
 88 nodes merged between parts        8 and        8
390 nodes were deleted by tolerancing
```

Up through 4000 parts can be merged under general tolerancing (i.e. no use of the **ptol** or **bptol** commands). 300 parts can be merged under special tolerancing (**ptol** and **bptol**).

The following is a common error to avoid. Suppose you create three parts that meet as shown in 140 and 141. Then define a sliding interface between parts 1 and 2 and also between parts 1 and 3. No nodes will be merged between parts 1 and 2 and between parts 1 and 3. However, nodes can be merged between parts 2 and 3. Sometimes you need to look closely in the graphics or carefully check the Merged Nodes Summary to detect this error. To fix this error, if indeed it is an error, use a dummy sliding interface between parts 2 and 3 to force no merging between those parts. Alternatively, use the **bptol** command with a negative number to avoid merging between those parts. You should also consider extending both interfaces 1 and 2 across to parts 3 and 2, respectively, because they may come in contact. This is an ambiguous situation since there are equally plausible situations where parts 2 and 3 should be merged together.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
sid 1 sv;
sid 2 sv;
block 1 3;1 3;1 3;
     1 2 1 2 1 2
sii -2;;;1 s;
sii ;-2;;2 s;
block 1 3;1 3;1 3;
     2.1 3 1 2 1 2
sii -1;;;1 m;
block 1 3;1 3;1 3;
     1 2 2.1 3 1 2
sii ;-1;;2 m;
merge
stp .2
```

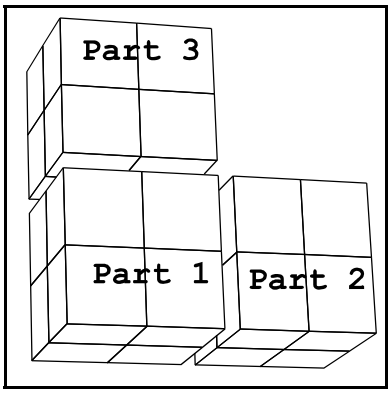


Figure 140 Before **stp**

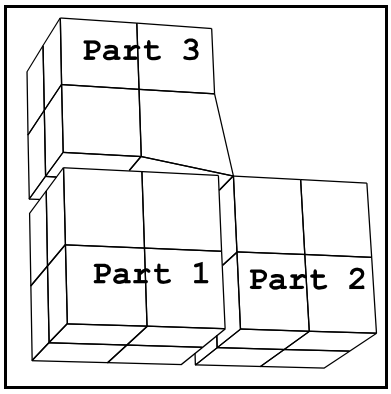


Figure 141 After **stp**

For other commands affecting how nodes are merged, see the "Merging Parts" section in the chapter "Global Commands".

mnl write the merged nodes list

mnl *(no arguments)*

Remarks

The merge table is written out to the save file. This may be useful when you are using **True*Grid*®** to get intermediate results and need to process the mesh data further. The table consists of two columns. The first column is the node number before merging. The second column is the node number after merging. Two rows with the same second node number means the two corresponding nodes have been merged together

Example

The following butterfly block structure is a good example of merging within one part.

```
sd 1 cy 0 0 0 0 0 1 3
block 1 2 3 4;1 2 3 4;-1;-1 -1 1 1 -1 -1 1 1 0
dei 1 2 0 3 4;1 2 0 3 4; -1;
sfi -1 0 -4;;-1;sd 1
sfi ;-1 0 -4;-1;sd 1
merge
stp .001
mnl
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

This produces the merged nodes table below. Notice that the original node number 5 is merged to node 1. This causes the original node 6 to be renamed as node 5. The original node 9 is merged to node 2, causing the original node 10 to be renamed node 8. The original node 11 is merged to node 5, which was originally node 6. The original node 12 is merged to node 8, which was originally node 10.

```
c MERGED NODES TABLE
c          1          1
c          2          2
c          3          3
c          4          4
c          5          1
c          6          5
c          7          6
c          8          7
c          9          2
c         10          8
c         11          5
c         12          8
```

pn place a node at a new location

pn *node x y z*

where

node	node number
x y z	Cartesian coordinates of the new location

Remarks

This command is used automatically when the *Move Pts.* and *node* button are selected in the Merge Phase environment window.

The node numbers change when a different merge command is issued. If two nodes are merged into one and assigned a new number, it is this number that is used in this command. If, after this node is moved using the **pn** command, a new merge command is issued with a negative value so that the two nodes are no longer merged together, both nodes will have been moved to the same location so that they are coincident but separate nodes.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

2. Diagnostics Commands

ajnp find node near point

ajnp *x_coordinate y_coordinate z_coordinate*

Remarks

The parameter, %node, is set to the node number of the node that is located nearest (Euclidean distance) the given point. This parameter is then available for future commands. This feature has been provided so that the node number can be used in batch mode.

cenref restore reference center for moment calculations

cenref *(no arguments)*

Remarks

Restore the reference center for moment calculations to its default, the origin of the coordinate system.

Normally, the origin of the coordinate system is used as a reference center in moment calculations. You can change that with **reference** command. The purpose of this command is simply to restore the default afterwards.

centroid moments and inertia

centroid *(no arguments)*

Remarks

Print the moments and inertias of each part and material.

elm highlight elements within a *measure* interval

elm *minimum maximum*
where

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

minimum	is the lower property limit, and
maximum	is the upper limit on the values of the property.

Remarks

This command is designed to be used after you have determined the quality level of your mesh using the **measure** command. First, you measure some property (orthogonality, aspect ratio, Jacobian, etc.) using the **measure** command. Then select a range of values for this property. The elements in that range will be highlighted. For example:

```
measure orthogon
elm -90 -45
```

highlights elements which contain angles less than 45 degrees. Notice that the orthogonality option of the measure reports the deviation from 90 degrees – not the absolute angles.

If, after merging nodes, an element is collapsed to form an illegal element (e.g. a 7 noded brick), many of the options in the **measure** command (from the volume, jacobian, stiffn, and pointvol options only) will report elements with a measure of -1.E10. To view these illegal elements set the minimum value in the **elm** command to the illegal element value. The orthogon, warp, and triangle option use -91 to indicate illegal elements are present.. The avolume, smallest, and aspect options use -1 to flag illegal elements.

elmoff turn off highlighting from the elm command

elmoff *(no arguments)*

info mesh model summary

info *(no argument)*

Example

The **info** command for the following input produced the table that follows.

```
title Spring Example
spd 1 le 1.1 spd 2 le 2.2 spd 3 dhpt 3.3 4.4
block 1 3 5;1 3 5;1 3 5;1 3 5;1 3 5;1 3 5;
spdp 1 1 3 2 2 3 1 m 1 dx dy dz;
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
spring 2 v1 3 3 3 dx1 sddn 3 amp 2.12;
npm 2 0 0 0 1.2 inc 2 dx rx;lct 1 rz 180;lrep 0 1;
block 1 3 5;1 3 5;1 3 5;1 3 5;1 3 5;6 8 10;
spdp 1 1 1 2 2 1 1 s dx dy dz;spring 2 v2 3 3 1 dx2;
lct 1 rz 180;lrep 0 1;merge
spring 1 n1 125 n2 335 rx1 ry1 rz1 rx2 ry2 rz2 sddn 2 amp 1;;
info

           2 parts
         502 nodes after merging
         256 linear solids
          18 bulk springs/dampers
           3 numbered springs/dampers
           2 point masses
```

mass mass table

mass *(no arguments)*

Print the masses, volumes and center of gravities of each part and material.

Remarks

The density of the material in this calculation is based on the material model. Be sure to define the material model before using this command. Densities are used only for materials defined for the following codes: ABAQUS, ALE3D, ANSYS, DYNA3D, ENIKE3D, ES3D, NASTRAN, LSDYNA, LSNIKE, MARC, NEUTRAL, NIKE3D, and TOPAZ3D.

measure choose a way to measure mesh quality at every element

measure *option*

where

option can be:

volume	to integrate the volume of each element,
avolume	to integrate the absolute volume,
jacobian	to compute the determinant of the Jacobian,
orthogon	to measure deviations from orthogonality (90 degrees) in quadrilaterals faces of elements,
smallest	for the smallest dimension of each element,
pointvol	to calculate the volume with a one point integration formula,

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

aspect	to calculate the aspect ratio for each element,
triangle	to measure deviations from optimum (60 degrees) in triangular faces of elements,
warp	to measure the angle between opposite corners of each element face
stiffn	stiffness or the condition number of the Jacobian
dvi	compare the volumes of neighboring elements in the i-direction - only for structured meshes,
dvj	compare the volumes of neighboring elements in the j-direction - only for structured meshes,
dvk	compare the volumes of neighboring elements in the k-direction - only for structured meshes,
dli	compare the lengths of neighboring edges in the i-direction - only for structured meshes,
dlj	compare the lengths of neighboring edges in the j-direction - only for structured meshes, and
dlk	compare the lengths of neighboring edges in the k-direction - only for structured meshes.

Remarks

A histogram is draw to show the profile of the mesh according to the selected measure. The abscissa is the measure and the ordinate is the number of elements when there is one measurement per element or element segments when there are several measurements per element. The range of the measurement is written to the save file and it is displayed in the text window during an interactive session. It is best to use this function after merging the nodes. Then illegal elements will be detected as well. This measure of quality will be restricted to those parts and materials showing in the graphics.

The volume option integrates the volume of a brick element using the tri-linear shape function to interpolate the volume. It is possible for the volume to be negative in some regions of the element; in that case the net volume will not be realistic. Shell elements are given thickness and the same method is then used to calculate the volume. If the shell element was not assigned a thickness, then the default of 1 is used. If an element is illegal, it will be assigned a measure of -1.E10.

The avolume option has the advantage that it is not affected by negative volumes since the absolute volume is integrated. Shells are given thickness as for the volume option above and then treated like a brick element. Illegal elements are flagged as -1.

The pointvol option approximates the volume of an element using the Jacobian at a single point in the center of the element. The shell elements are given thickness and treated the same as bricks (see

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

above). If an element is illegal, it will be assigned a measure of -1.E10.

The orthogonal option measures the 4 angles of all quadrilaterals in an element. It then graphs the deviation from 90 degrees. Illegal elements are flagged as -91 degrees.

Jacobian measures the shape of each element by sampling the Jacobian matrix of the map from the unit cube to the brick element at 27 Gauss points. In order to graph this data, the Jacobian matrix is reduced to a single number by first determining the eigenvalues of the matrix. The eigenvalue whose modulus is found between the other two is used to scale the Jacobian matrix. The matrix is divided by the cube of this modulus. The determinant of the resulting matrix is graphed. This is done to keep **True*Grid*®** dimensionless. Shell elements are given thickness to make this measurement. If an element is illegal, it will be assigned a measure of -1.E10.

The stiffn options measures the stiffness or condition number of the Jacobian at 27 Gauss points. Shell elements are given thickness to make this measurement.If an element is illegal, it will be assigned a measure of -1.E10.

The smallest option determines the smallest dimension of an element as the measurement. A brick element has 12 edges, 12 diagonals along the faces, and 4 interior diagonals. A shell element has 4 edges and 2 diagonals. Illegal elements are flagged as -1.

The warp option measures the angle between the normals at opposing nodes of each face. Illegal elements are flagged as -91 degrees.

The aspect ratio is defined as the ratio of the largest diagonal to the smallest diagonal of an element. A brick element has 12 diagonals along the faces and 2 interior diagonals. A shell element has 2 diagonals. If the largest diagonal is zero, then the ratio is set to zero. If the smallest diagonal is zero, the ratio is set to a very large constant. Illegal elements are flagged as -1.

The triangles option measures the 3 angles of all triangles in an element. It then graphs the deviation from 60 degrees. Illegal elements are flagged as -91 degrees.

With the **elm** and **elmoff** commands, you can see the locations in the mesh of the most interesting elements; e.g. you can use **measure** to measure volume and then **elm** to highlight the biggest elements.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

pmass active part mass

pmass *(no arguments)*

Remarks

The density of the material in this calculation is based on the material model. Be sure to define the material model before using this command. Densities are used only for materials defined for the following codes: ABAQUS, ALE3D, ANSYS, DYNA3D, ENIKE3D, ES3D, NASTRAN, LSDYNA, LSNIKE, MARC, NEUTRAL, NIKE3D, and TOPAZ3D.

reference reference point for moments and inertia

reference x_0 y_0 z_0

where

(x_0,y_0,z_0)	is the reference point

Remarks

This command changes the center reference point used in moment calculations. If you like, you can later issue the **cenref** command to restore this reference point to its default value: the origin of the coordinate system.

size dimensions of the mesh

size *(no arguments)*

smags detect detached small groups of elements

smags *element_set maximum*

Remarks

This command detects small detached groups of elements. These elements are put into a set. A cut off is specified and any detached group of elements with that or fewer elements will be placed into the garbage element set. This command is useful after reading a mesh with the **dynain** option of the **readmesh** command.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

tmass total mass

tmass *(no arguments)*

Remarks

The density of the material in this calculation is based on the material model. Be sure to define the material model before using this command. Densities are used only for materials defined for the following codes: ABAQUS, ALE3D, ANSYS, DYNA3D, ENIKE3D, ES3D, NASTRAN, LSDYNA, LSNIKE, MARC, NEUTRAL, NIKE3D, and TOPAZ3D.

3. Graphics Commands

backplane toggles back plane removal

backplane on
or
backplane off

Remarks

This affects the efficiency of the hidden line display graphics and the number of polygons in the draw. Specifically, it instructs the drawing algorithm to ignore all of those polygons which are "facing" towards the "back" of the image, thus greatly reducing re-draw time. This command is especially useful when re-drawing large, complicated models or when looking at all the IGES surfaces.

dpic dumps all picture parameters

dpic *(no arguments)*

Remarks

This is used to save a view to be read back into **True*Grid*®** using **rpic**.

rpic reads the picture parameters

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

rpic *picture_parameters*

Remarks

The picture parameters that were dumped by **dpic**.

4. Tokens in the Picture

This section describes tokens in the picture which identify features in the model. The **condition** command helps display boundary conditions, constraints, and the like. The **labels** command displays indices, surface numbers, and so forth. The **mlabs** command combines the **condition** and **labels** command and allows for multiple options.

Some of these labels and conditions may not be available in a filled (**tvv**) display. No tokens are available in OpenGL graphics (H.W. graphics).

mlabs multiple labels and conditions displayed

mlabs *options* ;

where an *option* can be any option available in the **labels** (**la**) or **condition** (**co**) commands.

Remarks

Any combination of options listed under both the labels and condition commands can be used in this command.

Example

```
mlabs nodes 3d ;
```

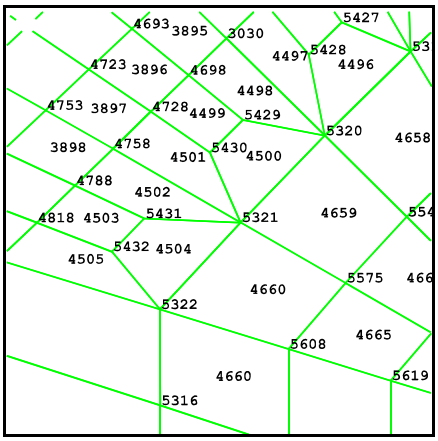


Figure 142 Multiple Labels

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

condition specify type of condition/constraint to be displayed

condition *option*
or
co *option*
where *option* can be:

		Relevant commands:
dx	for nodes with fixed translation in x	**b, bi**
dy	for nodes with fixed translation in y	**b, bi**
dz	for nodes with fixed translation in z	**b, bi**
rx	for nodes with fixed rotation in x	**b, bi**
ry	for nodes with fixed rotation in y	**b, bi**
rz	for nodes with fixed rotation in z	**b, bi**
mom *load_curve direction*	for nodes with prescribed torque	**mom, momi**
where *direction*		
x	around the x-axis	
y	around the y-axis	
z	around the z-axis	
fmom *load_curve*	follower moments	**fmom**
fc *load_curve*	for point load vectors at nodes	**ll, fc, fci**
fd *load_curve*	for forced displacement vectors at nodes	**fd, fdi, fdc, fdci, fds, fdsi**
pr *load_curve*	for pressure surface amplitude vectors	**pr, pri**
sy *symmetry_plane*	for nodes on symmetry planes	**plane**
where *symmetry_plane*	symmetry plane number	
si *sliding_interface type*	sliding interfaces or contact surfaces	**sid, si, sii**
where *sliding_interface*	sliding interface number	
type can be		
m	master	
s	slave	
b	both	
rb	for boundary radiation orientation vectors on faces	**rb, rbi**
re	for radiation enclosure orientation vectors on faces	**re, rei**
fl	for boundary flux orientation vectors on faces	**fl, fli**
cv	for boundary convection orientation vectors on faces	**cv, cvi, vcv, vcvi**
cvt	convection thermal loads	**cvt, cvti**
tm	for initial temperature at nodes	**tm, tmi, vtm, vtmi**

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

ft for prescribed temperature at nodes **ft, fti, vft, vfti**

fv *load_curve_number* for prescribed velocity at nodes **fv, fvi, fvc, fvci, fvs, fvsi, fvv, fvvi, fvvc, fvvci, fvvs, fvvsi**

sw *stone_wall_number* for nodes assigned to stone walls **plane, sw, swi**

nr for nonreflecting boundary faces **nr, nri**

jt *joint_number* for nodes in joints **jd, jt**

iss for interface save segments **iss, issi**

ve initial velocities **velocity, ve, vei**

efl for electric flux orientation vectors on faces **efl, efli**

vhg *load_curve* volumetric heat generation **vhg, vhgi, vvhg, vvhgi**

or sys to display element local coordinate system axis **or**

where

sys can be any one of the following:

r	to label the local element r-axis
s	to label the local element s-axis
t	to label the local element t-axis
rs	to label the local element r-axis and s-axis
tr	to label the local element r-axis and t-axis
st	to label the local element s-axis and t-axis
rst	to label the local element r-axis, s-axis, and t-axis

syf *plane* for nodes assigned to a symmetry plane with failure **plane, syf, syfi**

where

plane number of the symmetry plane with failure

bv prescribed boundary velocities **bv, bvi**

ol outlets **ol, oli**

il inlets **il, ili**

npb nodal print blocks **npb**

epb element print block **epb**

sp *spring/damper_#* springs and dampers **sp, spdp**

where a spring or damper number can be

0	for all numbered springs
n	for a numbered spring
-1	for all unnumbered springs

pm point masses **pm, npm**

interfac *slide_#* ABAQUS interface elements **sid, si, sii, bb**

thic shell thicknesses **th, thi, thic, ssf, ssfi**

mdep momentum deposition **mdep**

sfb *constraint* surface oriented constraints **sfb**

where

constraint can be:

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

dx	constraint in the local x-direction	
dy	constraint in the local y-direction	
dz	constraint in the local z-direction	
rx	constrain rotation about the local x-axis	
ry	constrain rotation about the local y-axis	
rz	constrain rotation about the local z-axuis	
tepro *load_curve*	temperature profile	**tepro**
sc	smoothing constraints	**sc**
acc *load_curve*	for prescribed acceleration	**acc, acci, accc, accci, accs, accsi, vacc, vacci, vaccs, vaccsi, vaccc, vaccci**
n	for shell element outward normal vectors	**n**
resn	for radiation enclosure surface numbers	**re, rei**
grtol *grid# patch#*	glued interfaces for CFX	**supblk, CFX4, stp, st, tp, p**

where

grid#	is the assigned part number for a grid
patch#	is the sequence number of the patch associated with that grid. The table of grid and patches is written to the screen each time the model is merged with either **stp**, or **tp** and this option is invoked. This is used only for the CFX output option. Be sure to merge the model before using this option or writing the output.

spw *spotweld_#*	numbered spotweld	**spw, spwd**
spwf *material_#*	Ls-dyna material 100 spotweld	**spwf**
ffc *load_curve*	follower point loads	**ffc**
bf *number*	bulk fluid	**bf, bfi, bfd**
frb *op load_curve*	prescribed nodal rotations	**frb**

where *op* can be

v	for velocity
a	for acceleration
d	for displacement (LS-DYNA only)
dofv	nodal DOF velocity
dofa	nodal DOF acceleration
dofd	nodal DOF displacement (LS-DYNA only)

detp	detonation points	**detp**
trp	tracer particles	**trp**
off	to turn off condition display	
size *scale*	to scale the size of the tokens displaying conditions	
angle θ	to change cone angle for tokens displaying conditions	

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

The tokens used to represent conditions on the model may be different when viewing the same condition in the wire and the hide mode graphics. For example, when viewing a sliding interface, the wire graphics circles every node on the interface. The hide shows an arrow at the center of every face on the interface, pointing in the outward direction. Both can be useful information.

co dx, dy, dz, rx, ry, or **rz** **display boundary conditions**

Example

co fc display concentrated nodal loads

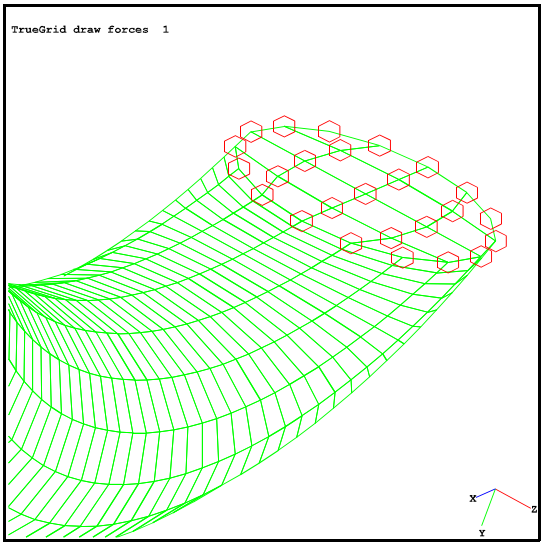


Figure 143 Wire frame version of CO FC

Example

```
cylinder 1 5;1 226;1 5;
    1.5 2.5 0 900 -.5 .5
sfi -1 -2;1 2;-1 -2;
     ts 0 0 0 0 0 1 2 0 1
z=z+0.05*j
fc 1 2 1 2 2 2 1
  1.2 .1 .2 .3
endpart
merge
co size 3.2
co fc 1
```

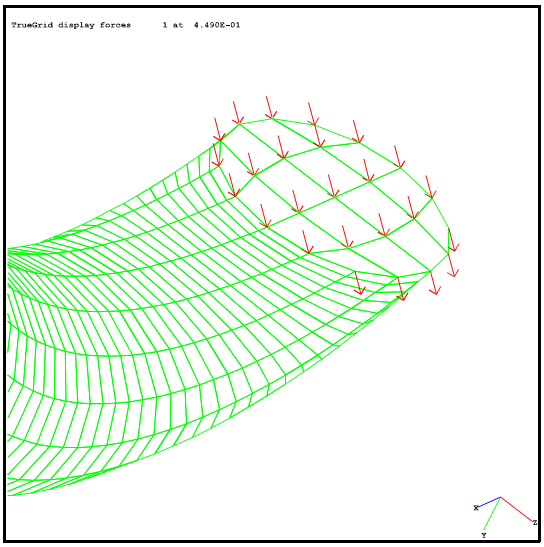


Figure 144 Hidden Line CO FC

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

The **co acc** (display accelerations), **co fmom** (display follower nodal moment), and **co fd** (display fixed displacements) options have a similar display.

co pr **display pressure**

Remarks

The **co bf** (display bulk fluid) is displayed in a similar fashion.**Example**

```
gct 1 mx 4 my 4 rz 15;
lev 1 grep 0 ;levct 11 rz 30;
    repe 11;;
lev 2 grep 0 1;;
pslv 1
pslv 2
block 1 4;1 4;1 4;5 7 5 7 5 7
sfi -1 -2;-1 -2;-1 -2;sp 6 6 6 1
lcd 1 0 0 1 1;
pr 2 1 1 2 2 2 1 1
endpart
pplv
pplv
merge
co pr 1 co size 3.2
```

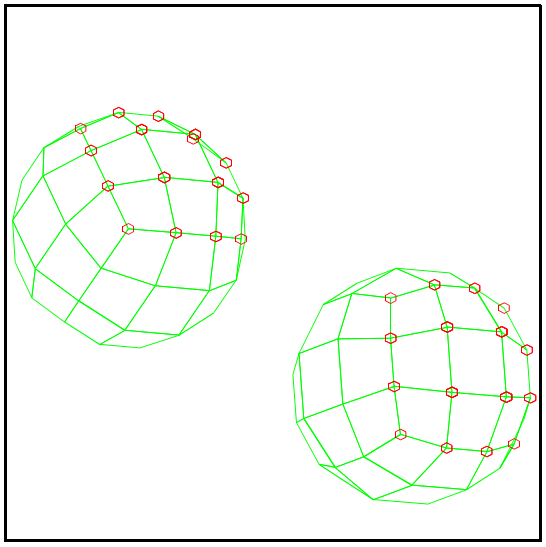


Figure 145 Wire frame version CO PR

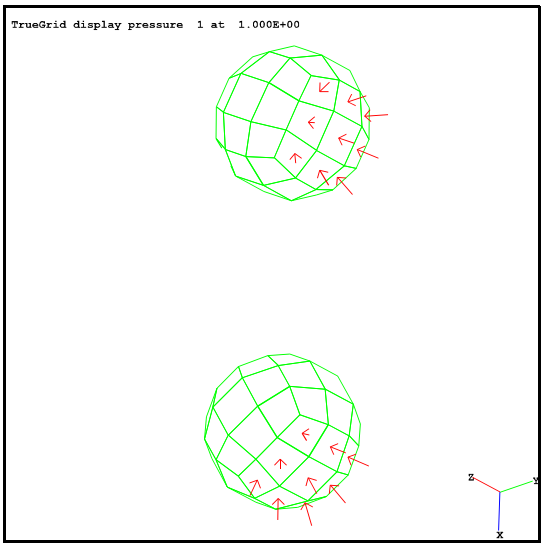


Figure 146 Hidden Line version CO PR

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

co si **display sliding interfaces**

Examples

co rb **display radiation boundary condition**

Remarks

To assign this condition, see the **rb**, **rbi**, **vrb**, and **vrbi** commands in the part phase and the **rb** command in the merge phase.

Example

```
block 1 3 5 7 9;
      1 3 5 7 9;
      1 3 5 7 9;
      -1 -1 0 1 1;
      -1 -1 0 1 1;
      -1 -1 0 1 1;
dei 1 2 0 4 5; 1 2 0 4 5;;
dei 1 2 0 4 5;;1 2 0 4 5;
dei ;1 2 0 4 5;1 2 0 4 5;
sfi -1 -5;-1 -5;-1 -5;sp 0 0 0 5
sfi -2 -4;-2 -4;-2 -4;sp 0 0 0 3
de 2 2 2 4 4 4
de 1 0 0 3 0 0
orpt - 0 0 0
rbi 3 -5;-1 -5;-1 -5;1 1 2 1
merge
co rb co size 3
```

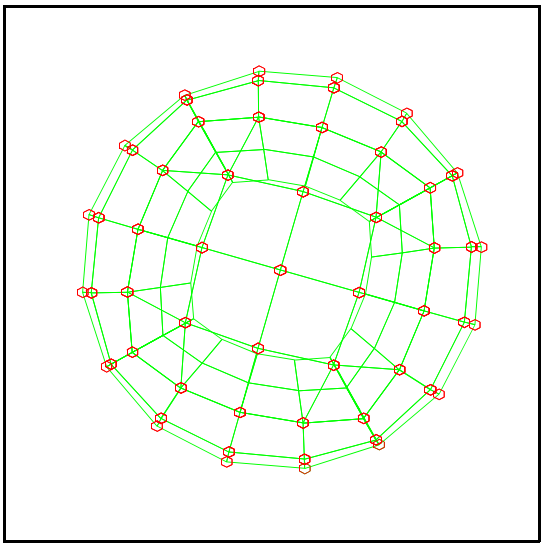


Figure 147 Wire frame version CO RB

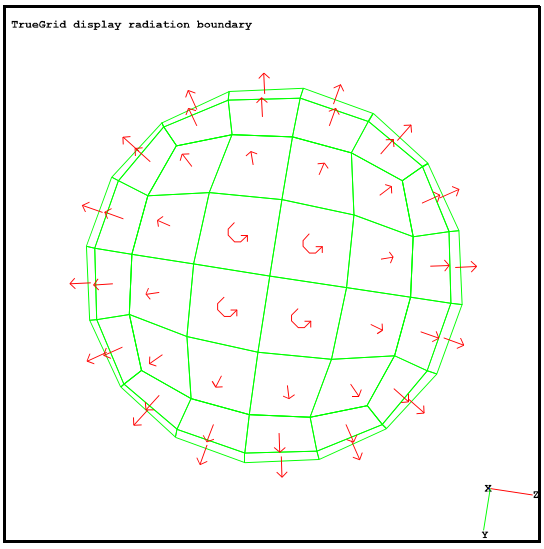


Figure 148 Hidden Line version CO RB

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
block 1 3 5 7 9;
      1 3 5 7 9;
      1 3 5 7 9;
      -1 -1 0 1 1;
      -1 -1 0 1 1;
      -1 -1 0 1 1;
dei 1 2 0 4 5;1 2 0 4 5;;
dei 1 2 0 4 5;;1 2 0 4 5;
dei ;1 2 0 4 5;1 2 0 4 5;
sfi -1 -5;-1 -5;-1 -5;sp 0 0 0 5
sfi -2 -4;-2 -4;-2 -4;sp 0 0 0 3
de 2 2 2 4 4 4
de 1 0 0 3 0 0
orpt + 0 0 0
rei 3 -4;-2 -4;-2 -4;0 2.2 no
merge
```

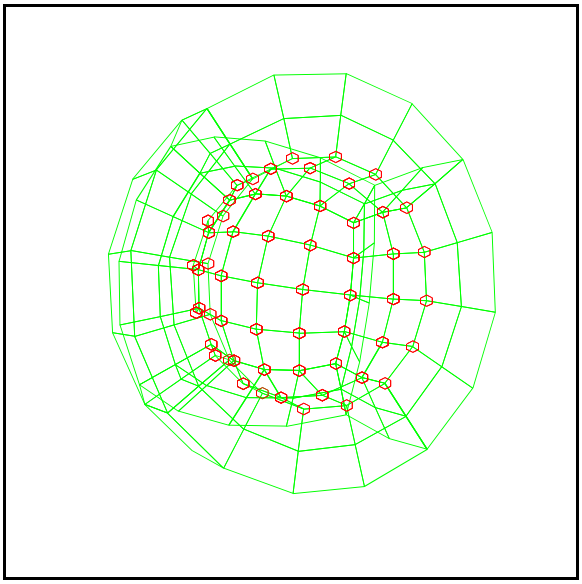


Figure 149 Wire frame version CO RE

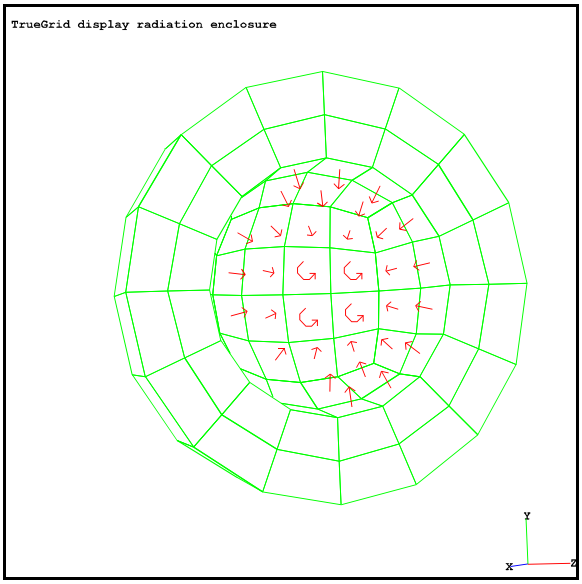


Figure 150 Hidden Line version CO RE

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
block 1 13 17 31 35 47;
      1 5 9 13;1 25;
      1 13 17 31 35 47;
      1 5 9 13;1 25;
dei 2 5; 1 2 0 3 4;;
mate 1
mti 1 3 ; 1 2 ; ; 2
mti 1 3 ; 3 4 ; ; 3
mti 4 6 ; 1 2 ; ; 4
mti 4 6 ; 3 4 ; ; 5
mbi -2; -1 0 -4; 1 2;x -1
mbi -5; -1 0 -4; 1 2;x 1
bb 2 2 1 3 2 2 1;
bi ;; -1;dz 1;
fdi -6;1 2 0 3 4;1 2;1 1 1 0 0
fdi -1;1 2 0 3 4;1 2;1 1 -1 0 0
lcd 1 0 0 1 1;
orpt - 24 7 13
fli -1 2 0 5 -6;-1 -4;-1 -2;1 1
merge
co fl co size 2.6
```

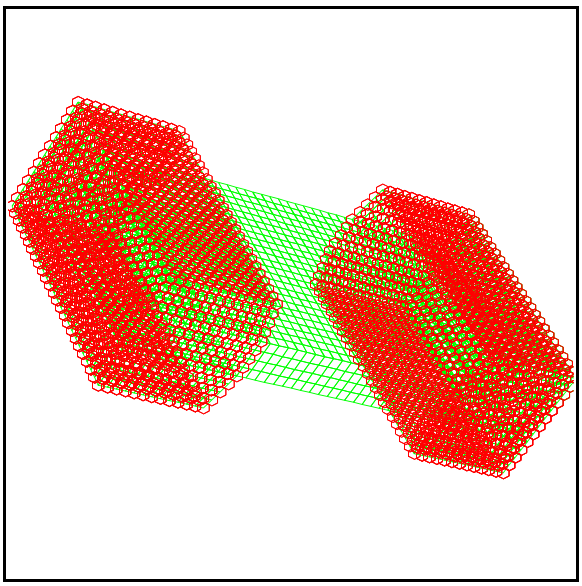


Figure 151 Wire frame version CO FL

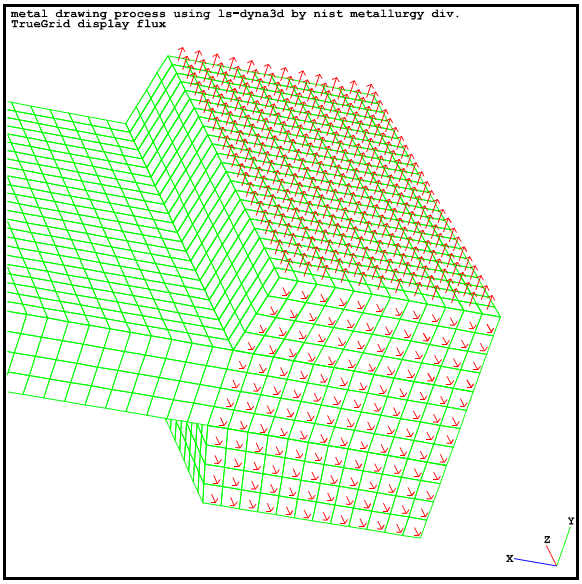


Figure 152 Hidden Line version CO FL

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

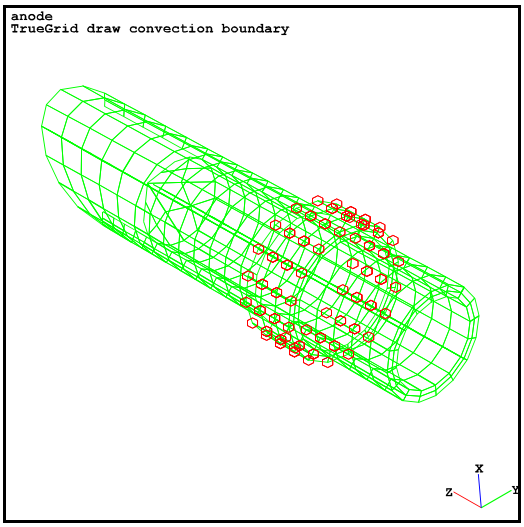


Figure 153 Wire frame version CO CV

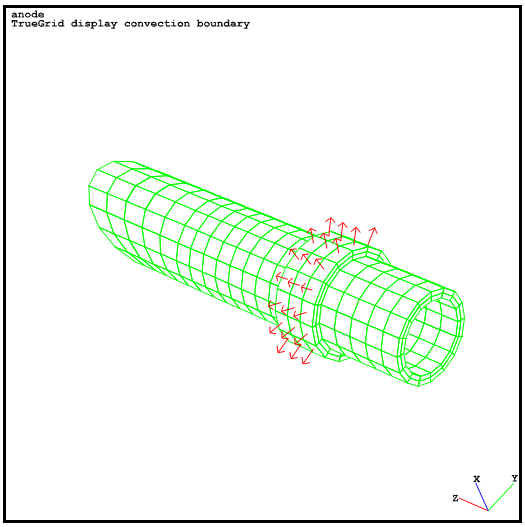


Figure 154 Hidden Line version CO CV

co cv **display convection conditions**
co cvt **display convection thermal loads**

Example

```
block -1 5 -9;-1 5 -9;-1 5 -9;-1 0 1;-1 0 1;-1 0 1;
sfi -1 -3 ;-1 -3 ;-1 -3 ; sp 0 0 0 5 de 1 0 0 2 0 0 orpt + 0 0 0
cvti 1 -3;-1 -3;-1 -3;.12 55 merge co cvt co size 2.4
```

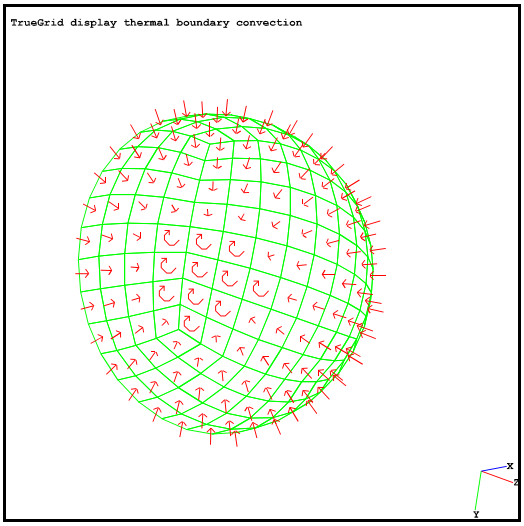


Figure 155 Hidden Line CO CVT

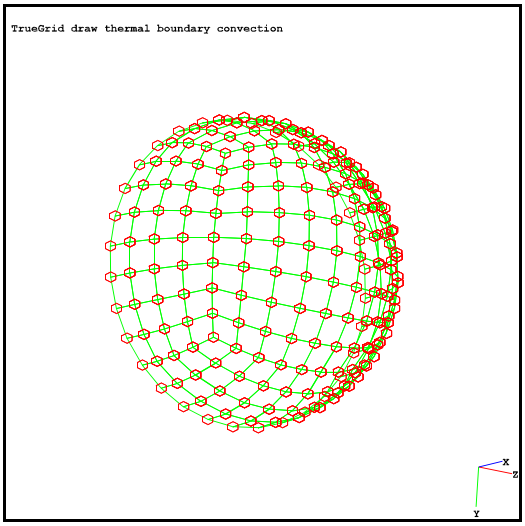


Figure 156 Wire frame CO CVT

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

co tm **display initial temperature**

Example

```
sd 1 cy 0 0 0 0 0 1 5
sd 2 cy 0 0 0 0 0 1 15
sd 3 plan 0 0 0 0 1 0
sd 4 plan 0 0 0 1 0 0
sd 5 plan 0 0 0 0 0 1
sd 6 plan 0 0 10 0 0 1
block 1 11;
      1 11;
      1 3;
      0 10 0 10 0 10
tr 1 2 1 2 2 2 my -10
               rz 90
               my 5 ;
mb 1 1 1 2 1 2 x 5
sfi -1; ; ;sd 1
sfi -2; ; ;sd 2
sfi ;-1; ;sd 3
sfi ;-2; ;sd 4
sfi ; ;-1;sd 5
sfi ; ;-2;sd 6
tfi ; ;-1 0 -2;
tf 1 1 1 2 2 2
tm 2 1 1 2 2 2 100
endpart
merge
rx 20 ry 20
co tm
co size 2
```

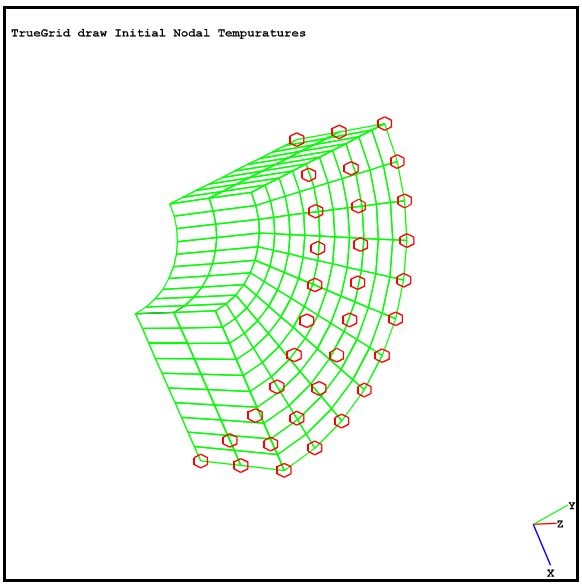


Figure 157 Wire frame version CO TM

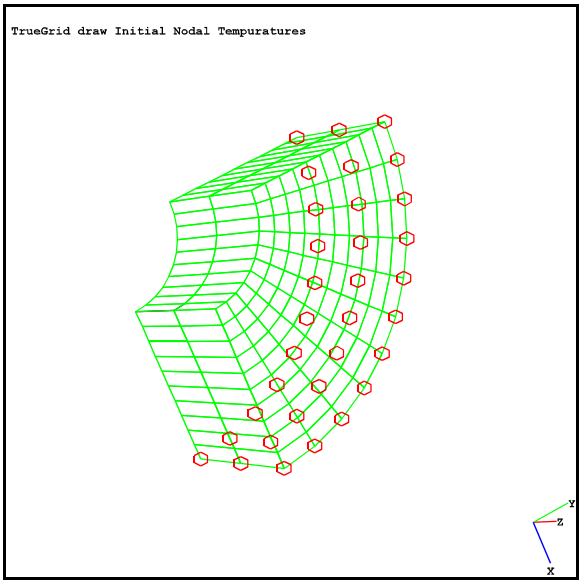


Figure 158 Hidden Line version CO TM

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

co ft **display boundary temp**

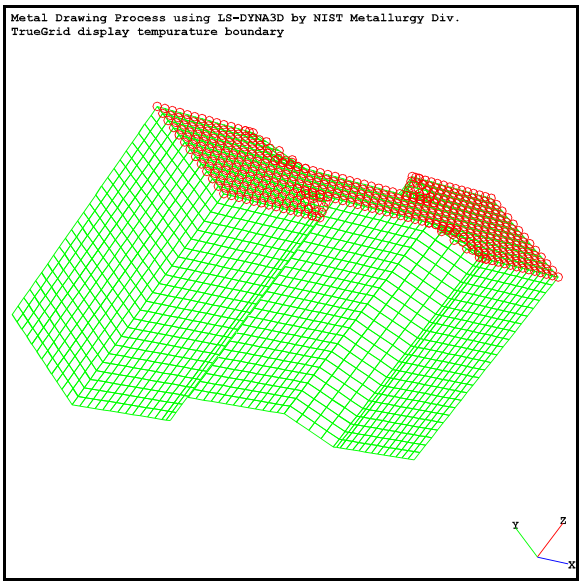


Figure 159 Hidden Line version CO FT

co fv **display nodal velocities**

Example

```
block 1 11; 1 11; 1 11; -1 1; -1
1; -1 1;
sfi -1 -2; -1 -2; -1 -2; sp 0 0 0
1
lcd 1 0 0 .1 1 1 1;
fv 2 1 2 2 1 2 1 1 .57 -.57 .57 fv 2 2 2 2 2 2 1 1 .57 .57 .57
```

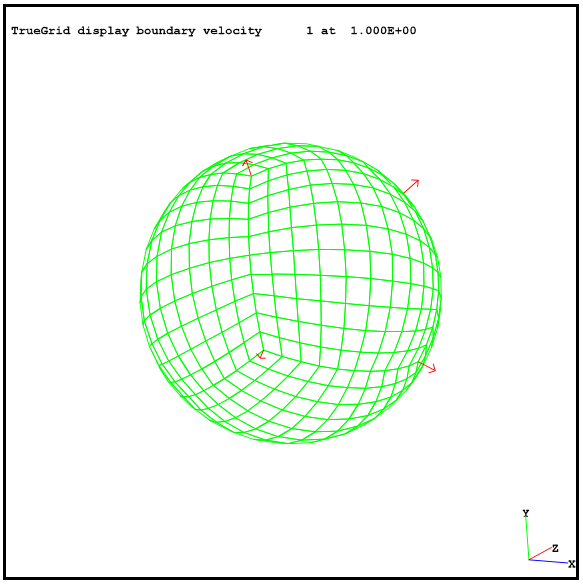


Figure 160 Hidden Line version CO FV

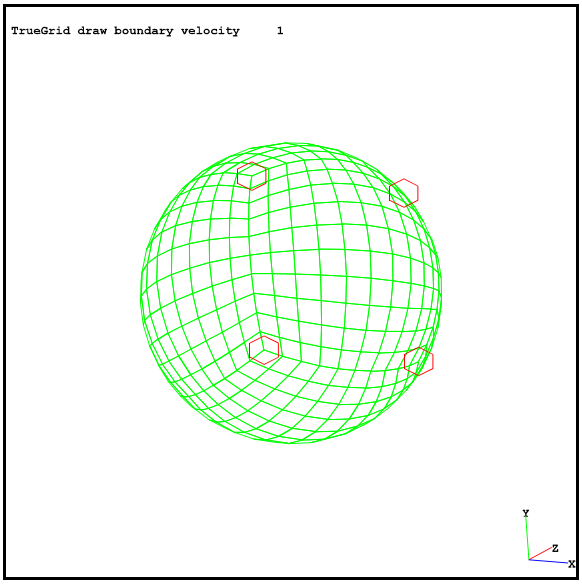


Figure 161 Wire frame version CO FV

```
fv 1 2 2 1 2 2 1 1 -.57 .57 .57 fv 1 1 2 1 1 2 1 1 -.57 -.57 .57
merge co fv 1 co size 2
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

co sw **display stone wall**

Example

```
sd 1 cy 0. 0. 3.8 1 0 0 1.
sd 2 cy 0. 0. 3.8 1 0 0 1.6
sd 3 cy 0. 0. 5.8 0 1 0 .25
sd 4 cy 0. 0. 5.8 0 1 0 .68
sd 5 cy 0. 0. 3.8 0 1 0 .125
block 1 4 6 8 11 14 17 19;1 3 5 7
9;1 6 12 14 18 20 22 26 28 31 33
35 37;-2.5 -1.9 -1.7 -1.5 -1 -.6
-.2 0 -1.5 -1 -.7 -.3 0 0 .7 2.2
3.15 3.65 3.8 3.95 4.45 5.4 5.6
5.8 6 6.2
dei 1 5;;2 13;;
dei 5 6;;2 3 0 9 13;
dei 7 8;;5 7 0 10 12;
dei ;1 2;2 13;dei ;3 5;4 8;
dei ;4 5;7 13;dei 2 4;3 5;1 2;
sfi ;-3 5;-4 -8;sd 1
sfi ;-2 5;-3 -9;sd 2
sfi ;-2;9 13;plane 0 -1.1 0 0 1 0
sfi ;-4;8 13;plane 0 -.08 0 0 1 0
sfi ;-2;2 3;plane 0 -1. 0 0 1 0
sfi ;-3;8 13;plane 0 -.6 0 0 1 0
sfi ;-3;3 4 ;plane 0 -.6 0 0 1 0
sfi ;-4;3 4 ;plane 0 -.3 0 0 1 0
sfi -7 8; ; -10 -12; sd 3
sfi -6 8 ; ; 11 -13; sd 4
sfi -7 8 ; ; -5  -7; sd 5
sfi -6;;8 11;plan -.68 0 0 1 0 0
sfi -2 -4;-3 4;;cy -1.7 -.187 0 0
0 1 .125
sfi -2;4 5;;plan -1.825 0 0 1 0 0
sfi -4;4 5;;plan -1.575 0 0 1 0 0
relaxi 6 8;-2;9 13;50 .001 1
relaxi 6 8;-4;9 13;50 .001 1
relaxi -8;2 5;3 9;50 .001 1
relaxi 5 8;-2;3 9;50 .001 1 relaxi
5 8;-3;4 8;50 .001 1
relaxi 1 6;1 5;-1;50 .001 1 relaxi 1 6;1 5;-2;50 .001 1
relaxi -5;2 5;3 9;50 .001 1 lct 3 ryz;rxz;ryz rxz;
lrep 0 1 2 3;endpart merge stp .001 nset n1 = l 997 998 1650 1651
1668 1669 2986 2987 3004 3005 3657 3658;sw nset n1 1
```

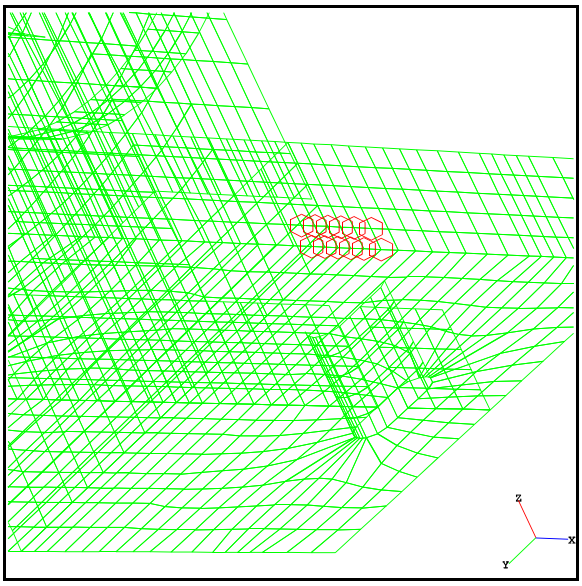


Figure 162 Wire frame version CO SW

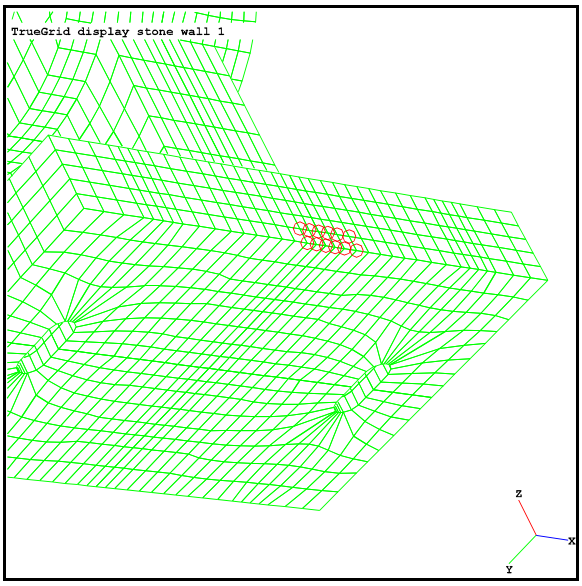


Figure 163 Hidden Line version CO SW

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

co nr **display non-reflecting**

Remarks

The **co iss** command works the same way.

Example

```
block 1 11;1 11;1 11;
      -1 1;-1 1;-1 1;
sfi -1 -2;-1 -2;-1 -2;sp 0 0 0 1
z=1.5*z
nri -1 -2;-1 -2;-1 -2;merge
```

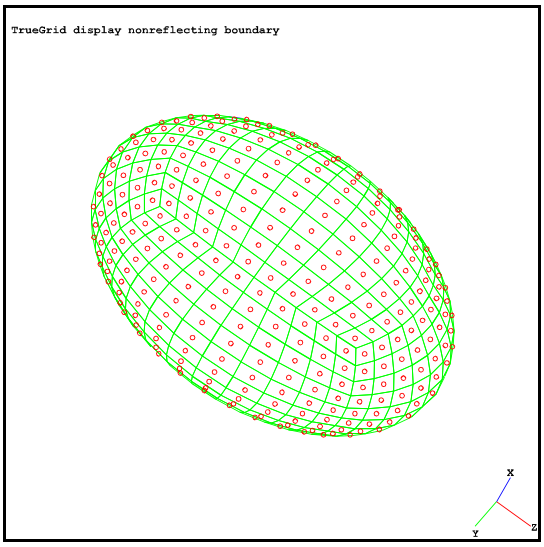


Figure 164 Hidden Line version CO NR

co ve **display initial velocities**

Example

```
block 1 3 5 7; 1 3 5 7;1 15;2 2 3
3;2 2 3 3;1 8;
sd 1 cyli 2.5 2.5 2.5 0 0 1 2 dei 1 2 0 3 4;1 2 0 3 4;;
sfi -1 -4; -1 -4;;sd 1 ve 1 1 2 4 4 2 .1 .1 1
```

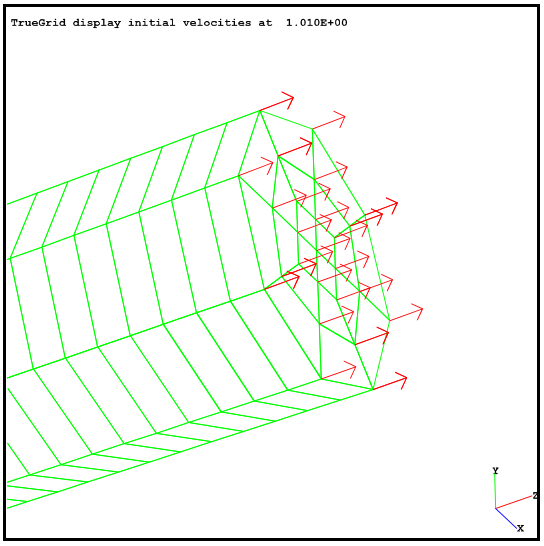


Figure 165 Hidden Line CO VE

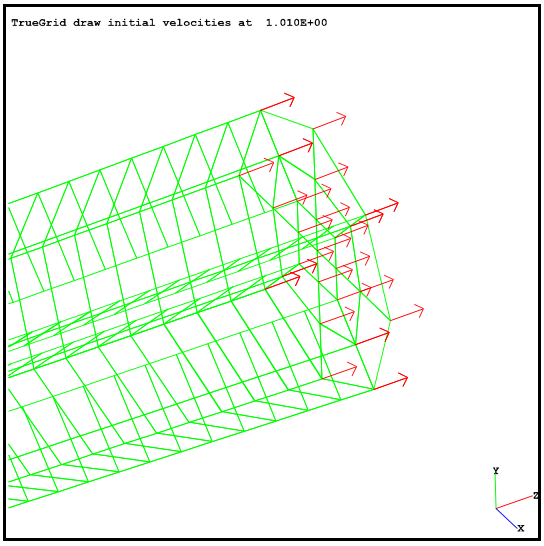


Figure 166 Wire frame CO VE

```
merge stp .0001 co ve co size 5
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

co efl **display electric flux**

Example

```
ld 1 lp2 1 1;
  lfil 45 1 2 -45 .25
  lp2 1 2;
sd 1 crz 1
cylinder -1;
         1 4 7 10 91;
         1 23;
         1;
         0 12 24 36 360;
         1 2;
sfi -1;; 1 2;sd 1
efl 1 1 1 1 2 2 1.3
efl 1 2 1 1 3 2 2.3
efl 1 3 1 1 4 2 1.3
endpart
merge
rx -30 ry 30 zf 1.3 tvv
co efl co size 2.8
```

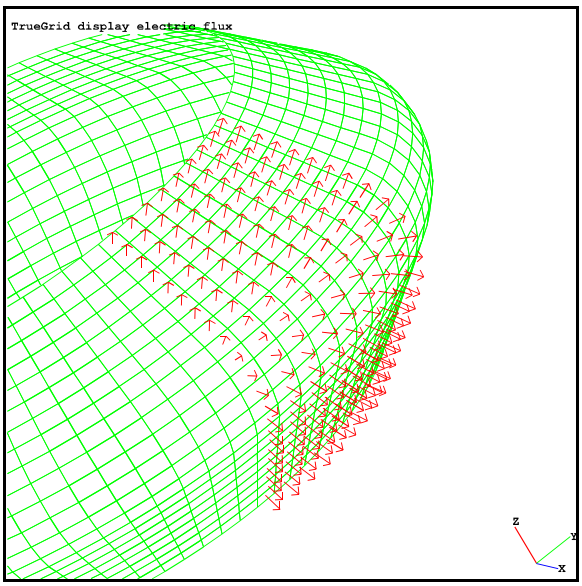


Figure 167 Hidden Line version CO EFL

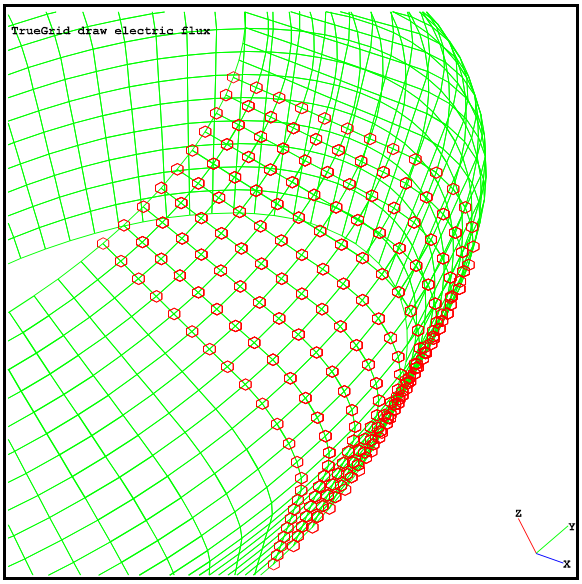


Figure 168 Wire frame version CO EFL

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

co vhg **display volumetric heat generation**

Example

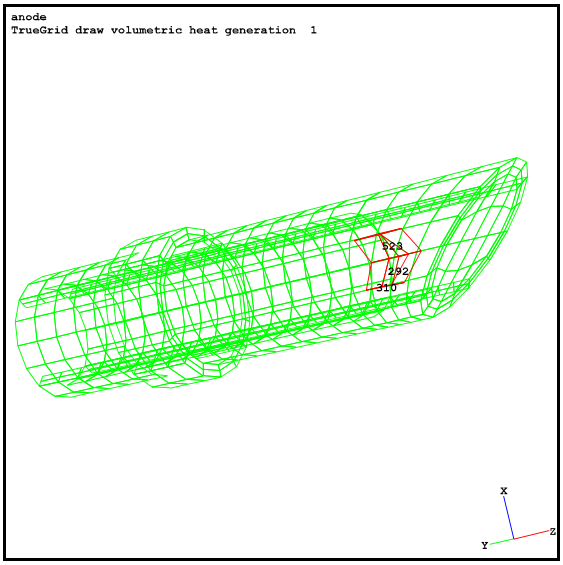


Figure 169 Wire frame version CO VHG

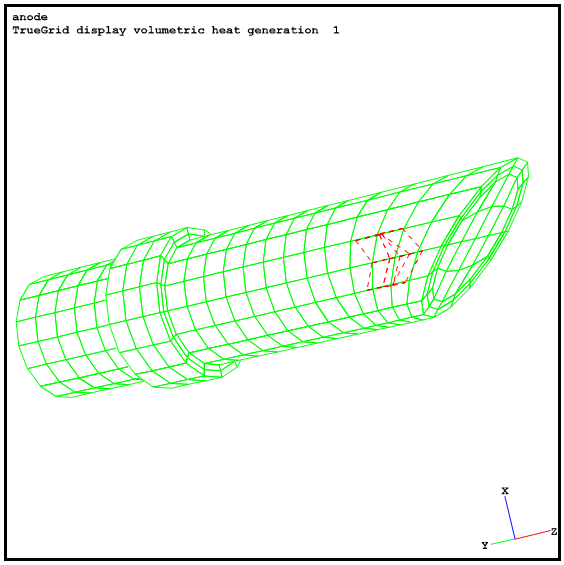


Figure 170 Hidden Line version CO VHG

This example is based on the anode.tg batch file in the EXAMPLES directory.

co sfb **display surface boundary conditions**

Remarks

The **co or** options works in a similar fashion.

Example

```
ld 1 csp2 00 1 -1 1.2 0 1 1; ;
sd 1 crz 1
cylinder -1;1 6;1 10;
         1 -30 30 -1 1
sfi -1;;;sd 1
sfb 1 1 1 1 2 2 surface j dy 1 ;
merge
```

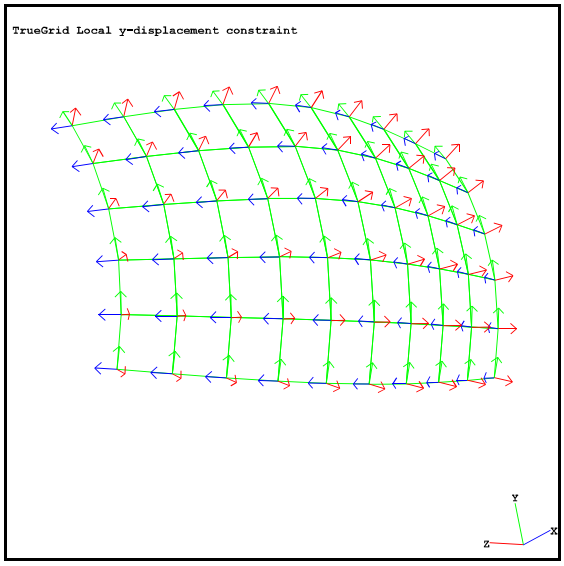


Figure 171 Wire frame version CO SFB

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

co syf **display symmetry plane with failure**

Example

```
sd 1 sp 0 0 0 5
sd 2 sp 0 0 0 5.5
block 1 3 5 7 9 11 13;
      1 3 5 7 9 11 13;
      1 3 5 7 9 11 13;
  -2.5 -2.5 -2.5 0 2.5 2.5 2.5;
  -2.5 -2.5 -2.5 0 2.5 2.5 2.5;
  -2.5 -2.5 -2.5 0 2.5 2.5 2.5;
dei ;1 3 0 5 7; 1 3 0 5 7;
dei 1 3 0 5 7;; 1 3 0 5 7;
dei 1 3 0 5 7; 1 3 0 5 7;;
sfi -2 -6;-2 -6;-2 -6;sd 1
sfi -1 -7;-1 -7;-1 -7;sd 2
sfi -4;;;plan 0 0 0 1 0 0
de 1 0 0 4 0 0
plane 1 0 0 0 1 0 0 .0001 syf
syf 4 1 1 4 7 7 1 .9
merge
```

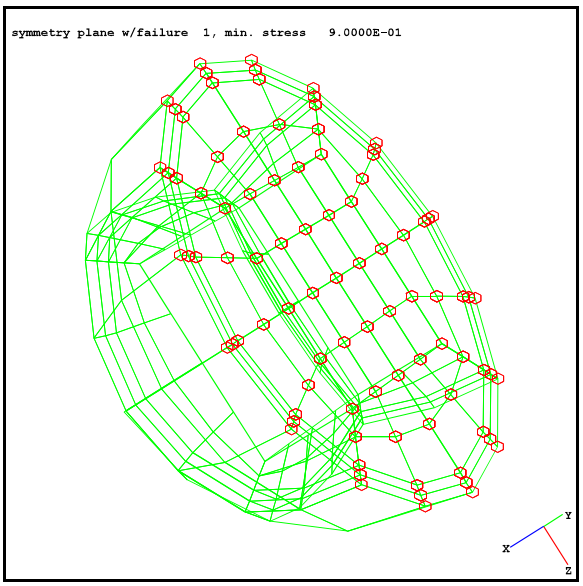


Figure 172 Wire frame version CO SYF

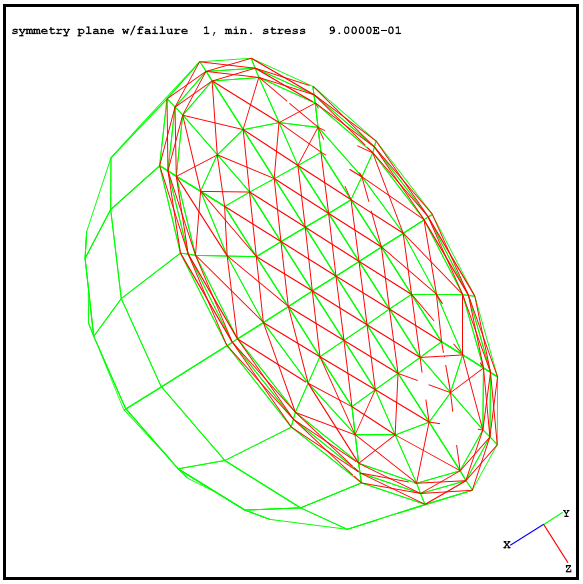


Figure 173 Hidden Line version CO SYF

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

co bv **display boundary velocities**

Example

```
cyli 1 3 5 7 9;
     1 41;
     1 3 5 7 9;
     1 2 3 4 5;
     0 360;
     0 1 2 3 4;
bv 1 1 5 5 2 5 .3 0 .5
b 1 1 1 5 2 1 dx 1 dy 1 dz 1 ;
merge
co bv co size 2.9
```

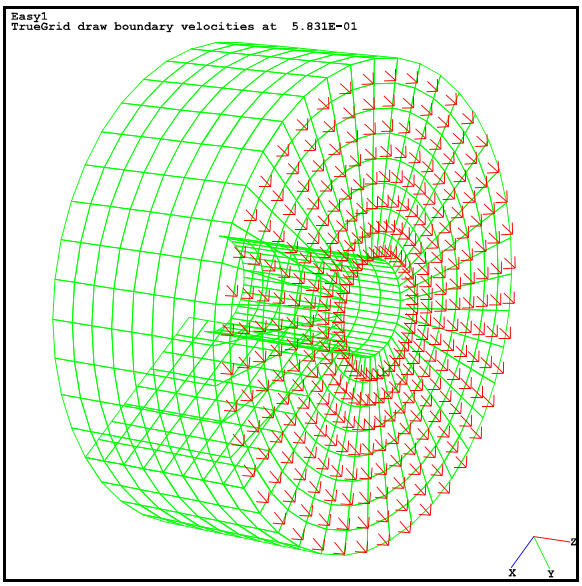


Figure 174 Wire frame version CO BV

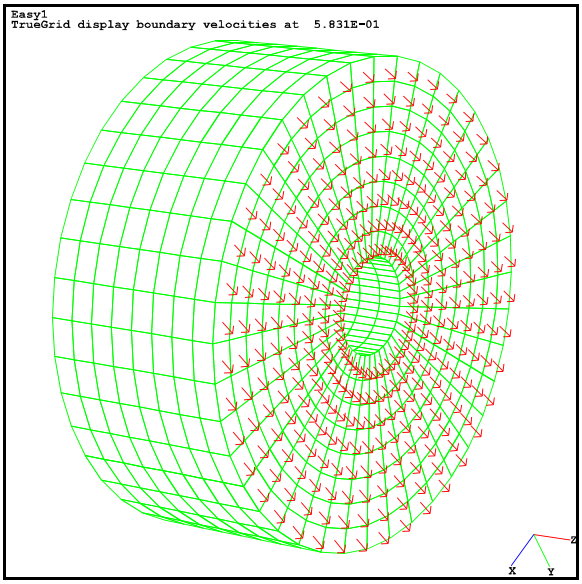


Figure 175 Hidden Line version CO BV

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
sd 1 sp 0 0 0 5
sd 2 plan 0 0 3 0 0 1
sd 3 plan 0 0 -3 0 0 1
block -1 -5 -9;
      -1 -5 -9;
       1 5 9;
       -1 0 1;
       -1 0 1;
       -1 0 1;
sfi -1 -3 ;-1 -3 ;1 3 ; sd 1
sfi ;; -3;sd 2
sfi ;; -1;sd 3
merge
nset nodes1 = l 112 117 200 253
268;
npb nset nodes1
```

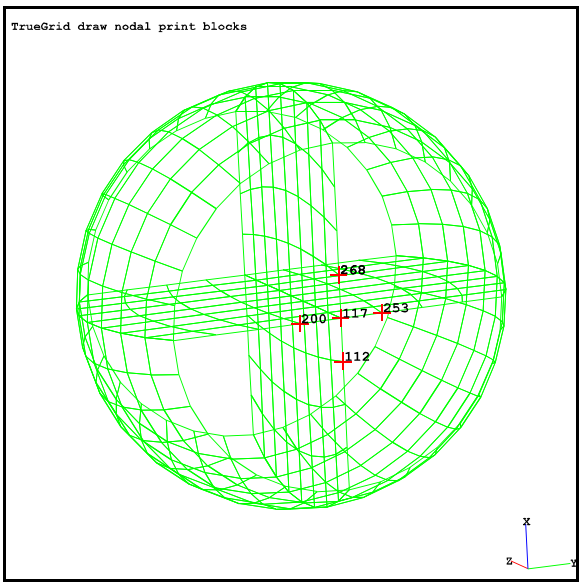


Figure 176 Wire frame version CO NPB

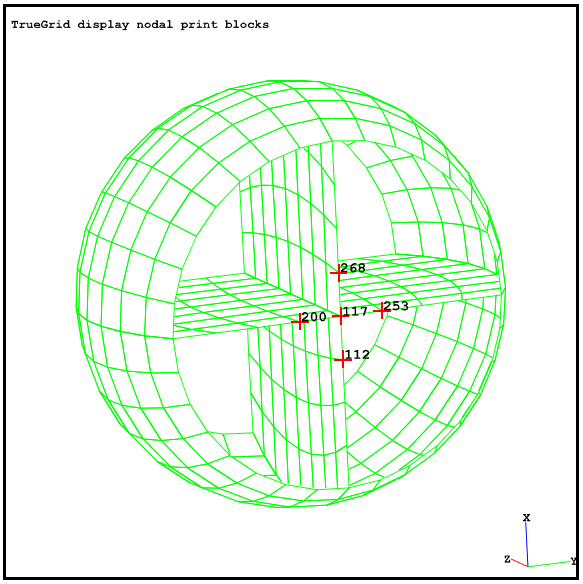


Figure 177 Hidden Line version CO NPB

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
sd 1 sp 0 0 0 5
sd 2 plan 0 0 3 0 0 1
sd 3 plan 0 0 -3 0 0 1
block -1 -5 -9;-1 -5 -9;1 5 9;
       -1 0 1;-1 0 1;-1 0 1;
sfi -1 -3 ;-1 -3 ;1 3 ; sd 1
sfi ;; -3;sd 2
sfi ;; -1;sd 3
merge
eset elements1 rpl ls
 4 20 65 132 177 289 324 353;
epb s elements1
```

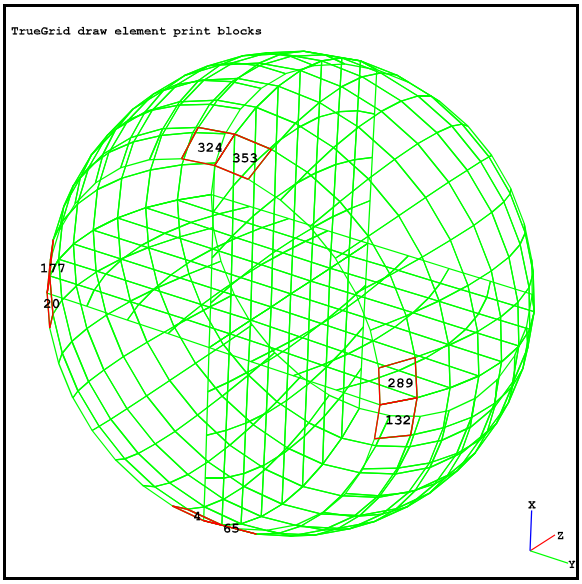


Figure 178 Wire frame version CO EPB

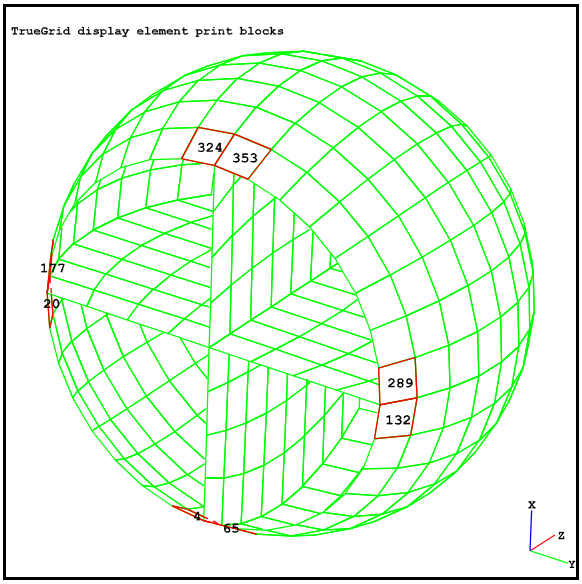


Figure 179 Hidden Line version CO EPB

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
cylinder 1 3;1 226;1 3;
   1.5 2.5 0 900 -.5 .5
sfi -1 -2;1 2;-1 -2;ts 0 0 0 0 0 1
2 0 1
z=z+0.05*j
jbm 1 1 1 2 2 2 1 1 1 i 1 ;
mate 0
merge
spring 1 n1 1 n2 448 sddn 1 ; ;
```

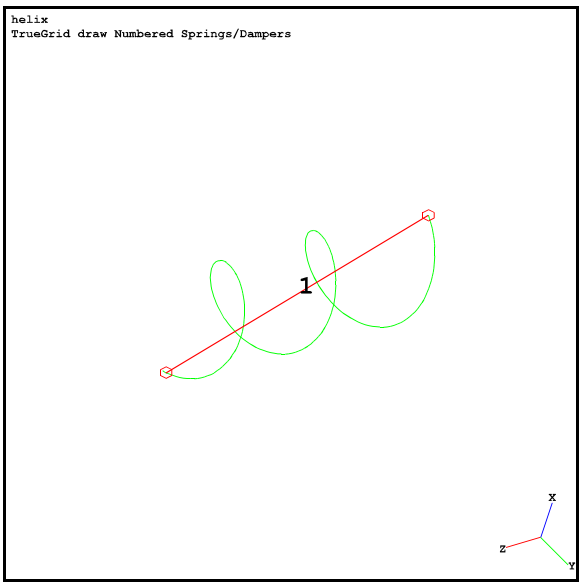


Figure 180 Wire frame version CO SP

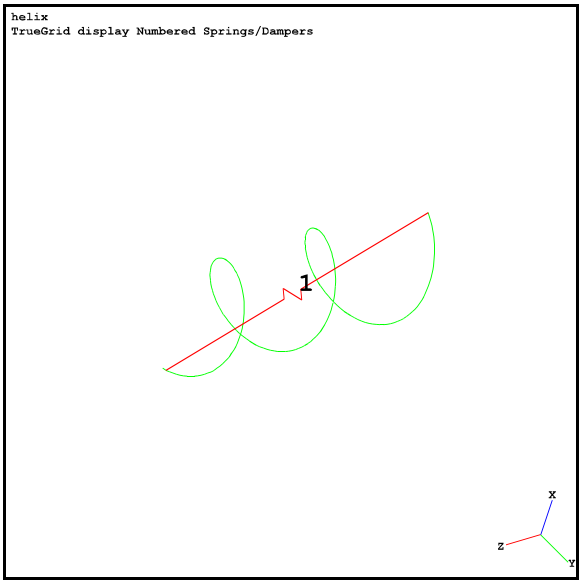


Figure 181 Hidden Line verison CO SP

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
sd 1 cy 0 0 0 0 0 1 1
sd 2 cy 0 0 0 0 1 0 1
sd 3 plan 0 0 0 0 1 0
sd 4 plan 0 0 0 0 1 1
sd 5 plan 0 0 0 0 1 -1
block 1 3 5 7;1 3 5 7;1 3;
-.3 -.3 .3 .3 -.3 -.3 .3 .3 1 2
dei 1 2 0 3 4; 1 2 0 3 4;;
sfi -1 -4; -1 -4;;sd 1
insprt 1 4 2 1
sfi ; -3;;sd 3
sfi -1 0 -4;; -1;sd 2
sfi 2 3; -1 0 -5; -1;sd 2
sfi ; 1 3; -1;sd 4
sfi ; 3 5; -1;sd 5
sd 6 plan .3 -.3 2 0 1 0
sd 7 plan -.3 .3 2 0 1 0
sd 8 plan .3 -.3 2 1 0 0
sd 9 plan -.3 .3 2 1 0 0
sfi -2; 2 4;;sd 9
sfi -3; 2 4;;sd 8
sfi 2 3; -2;;sd 6
sfi 2 3; -4;;sd 7
mseq i 4 6 4
mseq j 4 3 3 4
mseq k 6
lct 3 rx 90;rx 180;rx 270;
lrep 0 1 2 3 ;
pm 1 3 2 1 3 2 1 ;
endpart
merge
stp .001
pm 37 2 ;
npm 1 -2 0 0 2 ;
bm n1 37 n2 9262 ;
```

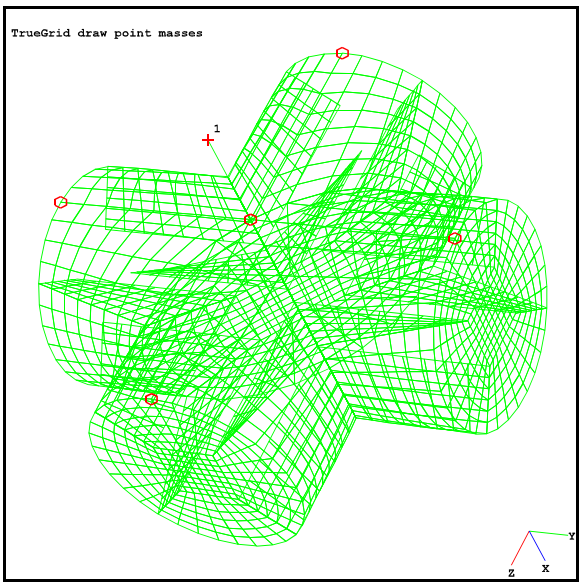


Figure 182 Wire frame version CO PM

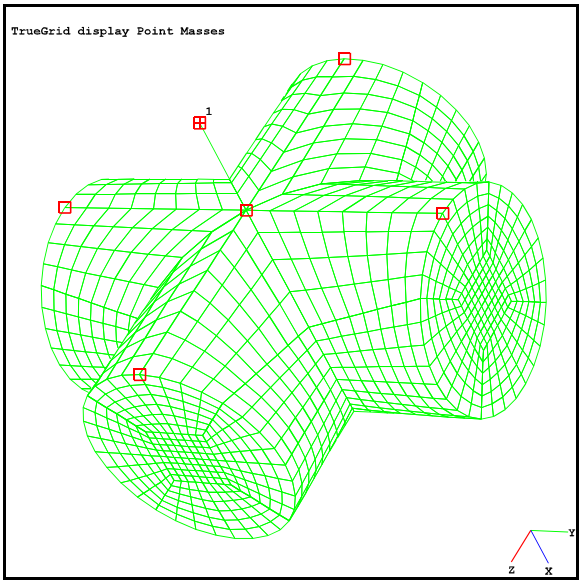


Figure 183 Hidden Line version CO PM

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
sd 1 function -1 1 -1 1 u+v ;
              u-v ;
              (u+v)*(u-v) ; ; ;
sid 1 inter;
block 1 11;1 11;1 3 4;
     -1 1 -1 1 -1 1 1
tr 1 1 1 2 2 3 rz 45;
patch 1 1 2 2 2 2 1
bb 1 1 2 2 2 2 1;
bb 1 1 1 2 2 1 1 mz .5;
bb 1 1 3 2 2 3 1 normal .3;
si 1 1 2 2 2 2 1 m ;
mti ;;2 3;2
block 1 11;1 11;1 3;-1 1 -1 1 1 5
tr 1 1 1 2 2 2 rz -45;
bb 1 1 1 2 2 1 1 normal .3;
bb 1 1 2 2 2 2 1 normal .9;
si 1 1 1 2 2 1 1 s ;
merge
```

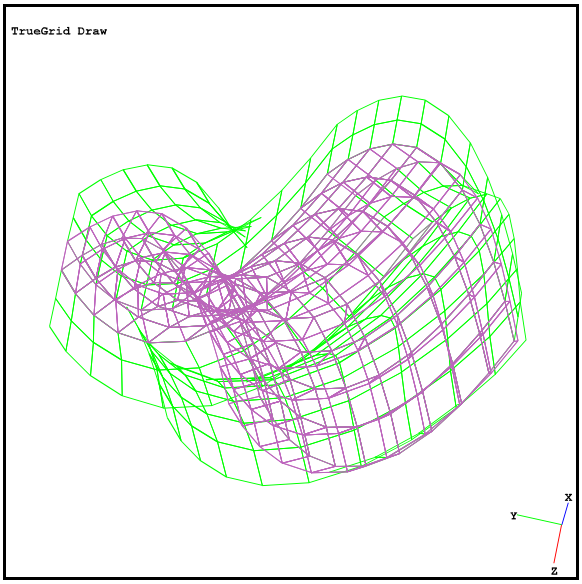


Figure 184 Wire frame version CO Interfac

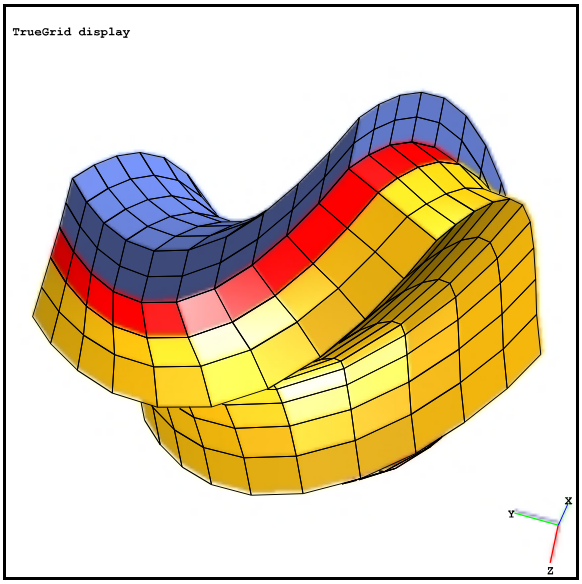


Figure 185 Fill verison CO Interfac

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

co thic **display shell thicknesses**

Example

```
ld 1 lep 1 2 3 0 110 180 0 ;
sd 1 crz 1
sd 2 cy 0 0 0 0 0 1 1.9
sd 3 intp 1 2 .5 ;
block -1 -11;-1 -11;1 6;
          -2 2 -2 2 0 1
ssfi -1 -2; -1 -2;;3
edge 1 1 1 2 1 1 3.3
edge 1 2 1 2 2 1 3.3
edge 2 1 1 2 2 1 3.3
edge 2 1 1 2 2 1 3.3
edge 2 1 2 2 2 2 3.1
edge 1 1 2 1 2 2 3.1
edge 1 1 2 2 1 2 3.1
edge 1 2 2 2 2 2 3.1
merge
```

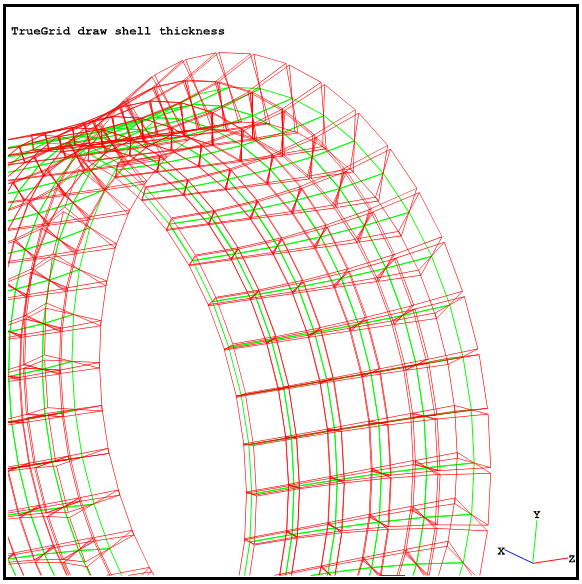


Figure 186 Wire frame version CO THIC

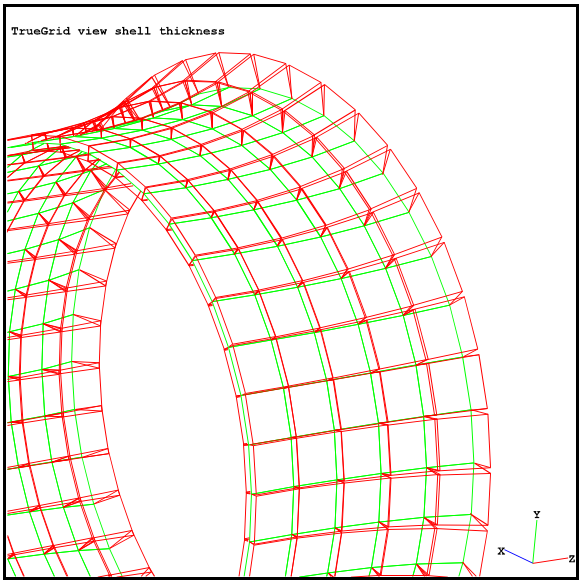


Figure 187 Hidden Line version CO THIC

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
sd 1 sp 0 0 0 1
sd 2 sp 0 0 0 2
sd 3 plan 0 0 0 -1 0 1
sd 4 plan 0 0 0 1 0 1
sd 5 plan 0 0 0 -1 1 0
sd 6 plan 0 0 0 1 1 0
block 1 5;1 5;1 5;
      1 2 -1 1 -1 1
sfi -1;;;sd 1
sfi -2;;;sd 2
sfi ;;-2;sd 3
sfi ;;-1;sd 4
sfi ;-2;;sd 5
sfi ;-1;;sd 6
lcd 1 0 0 1 1;
tepro   1   1   1   2   2   2   1
1.5*sqrt((x-2)^2+y*y+z*z)    ;
atan2(y,x) ;
merge
```

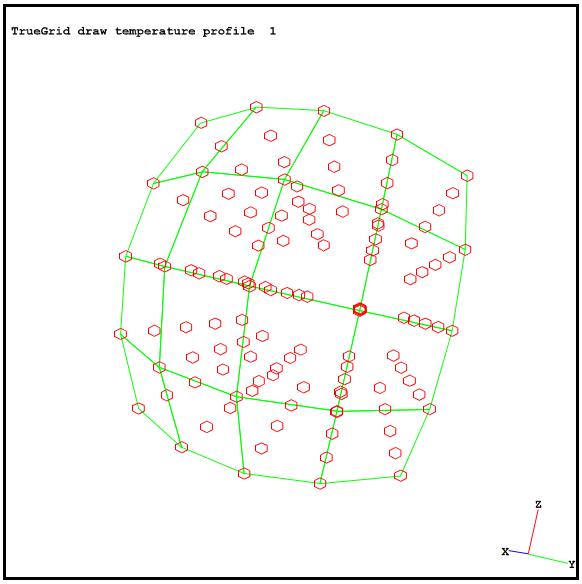


Figure 188 Wire frame verison CO TEPRO

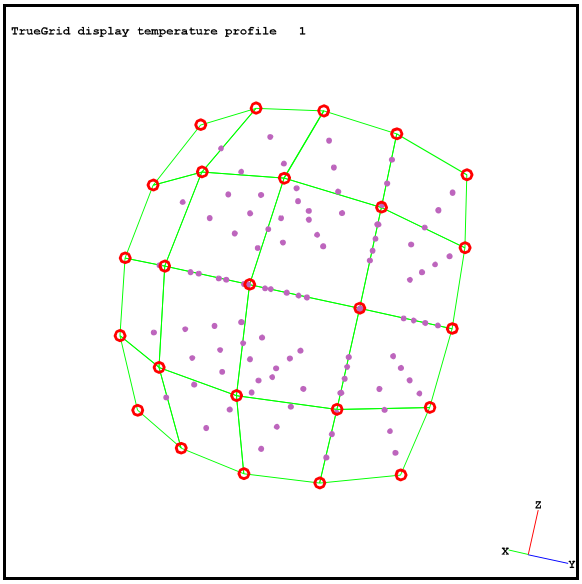


Figure 189 Hidden Line version CO TEPRO

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
sd 1 cy 0 0 0 0 0 1 2
block -1 2 5 -6;1 4;1 4;
      -1 -1 1 1 -1 1 -1 1
sfi -1 0 -2 0 -3 0 -4;;;sd 1
sc 2 1 1 3 2 2 i
merge
```

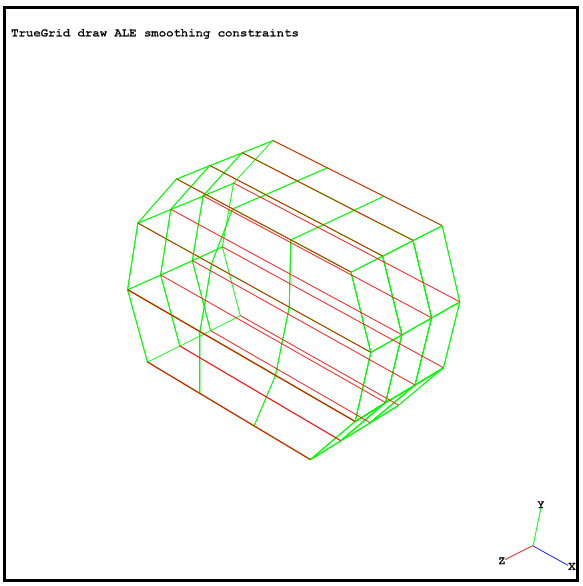


Figure 190 Wire frame version CO SC

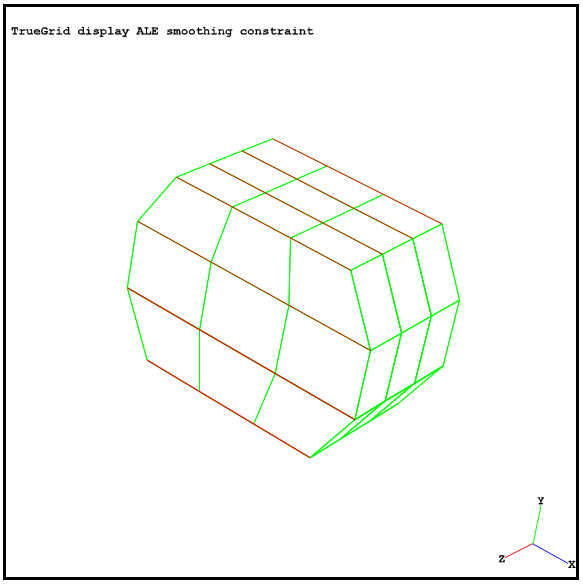


Figure 191 Hidden Line version CO SC

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

co n **display shell normals**

Example

```
block -1 -6;-1 -6;-1 -6;
      -1 1 -1 1 -1 1
sd 1 sp 0 0 0 1
sfi -1 -2; -1 -2; -1 -2;sd 1
orpt - 0 0 0
n 1 1 1 1 2 2
n 2 1 1 2 2 2
n 1 1 1 2 1 2
n 1 2 1 2 2 2
n 1 1 1 2 2 1
n 1 1 2 2 2 2
merge
```

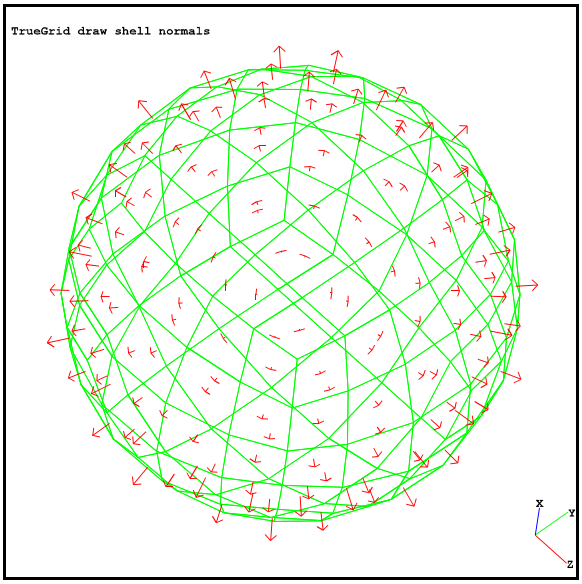


Figure 192 Wire frame version CO N

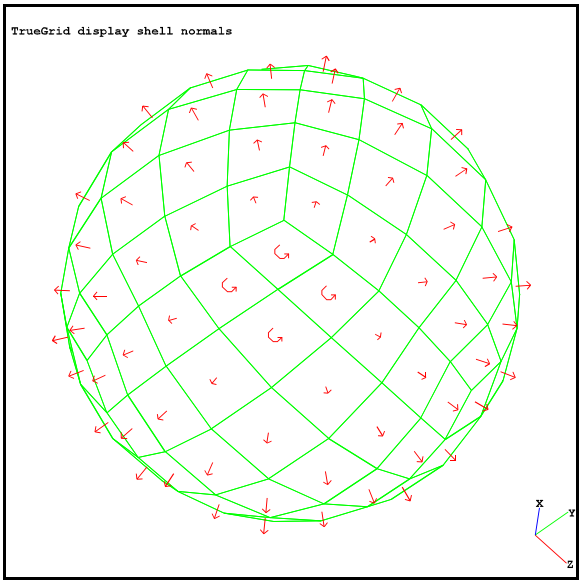


Figure 193 Hidden Line version CO N

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

co resn **display radiation enclosures labeled**

Example

```
block 1 3 5 7 9;
      1 3 5 7 9;
      1 3 5 7 9;
      -1 -1 0 1 1;
      -1 -1 0 1 1;
      -1 -1 0 1 1;
dei 1 2 0 4 5; 1 2 0 4 5;;
dei 1 2 0 4 5;;1 2 0 4 5;
dei ;1 2 0 4 5;1 2 0 4 5;
sfi -1 -5;-1 -5;-1 -5;sp 0 0 0 5
sfi -2 -4;-2 -4;-2 -4;sp 0 0 0 3
de 2 2 2 4 4 4
de 1 0 0 3 0 0
orpt  + 0 0 0
rei 3 -4;-2 -4;-2 -4;0 2.2 no
merge
```

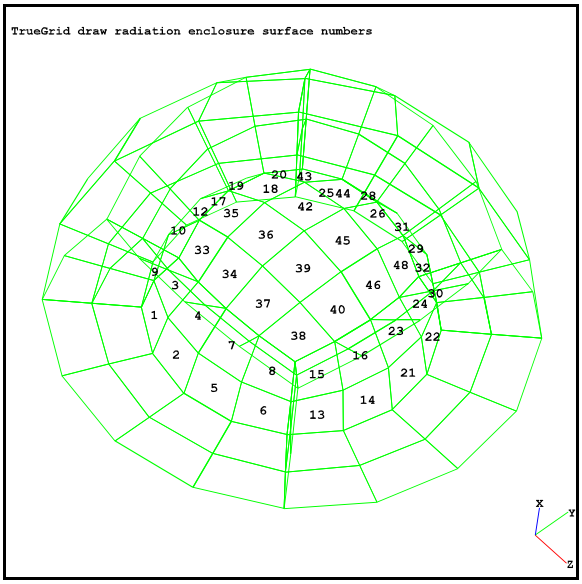


Figure 194 Wire frame version CO RESN

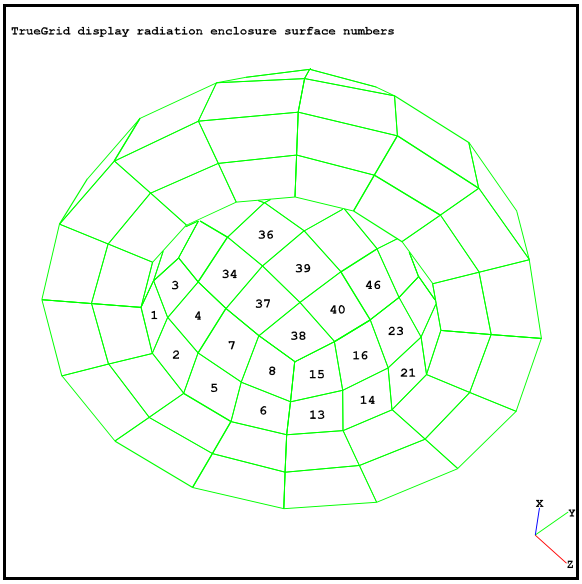


Figure 195 Hidden Line version CO RESN

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

co grtol **display glued interface for CFX**

Remarks

See the **supblk** command.

Example

A butterfly mesh of the fluid in the pipe is created. The **supblk** command was used to define grid 1. The interface 2 for grid 1 was displayed by the **co grtol** command (**Figure 196**). The simplified command file follows:

... transformations, projections of mesh ...

```
supblk 1 1 2 3 4 3
```

```
merge
```

... display of part 1...

```
co grtol 1 2   c display interface 2 for grid 1
```

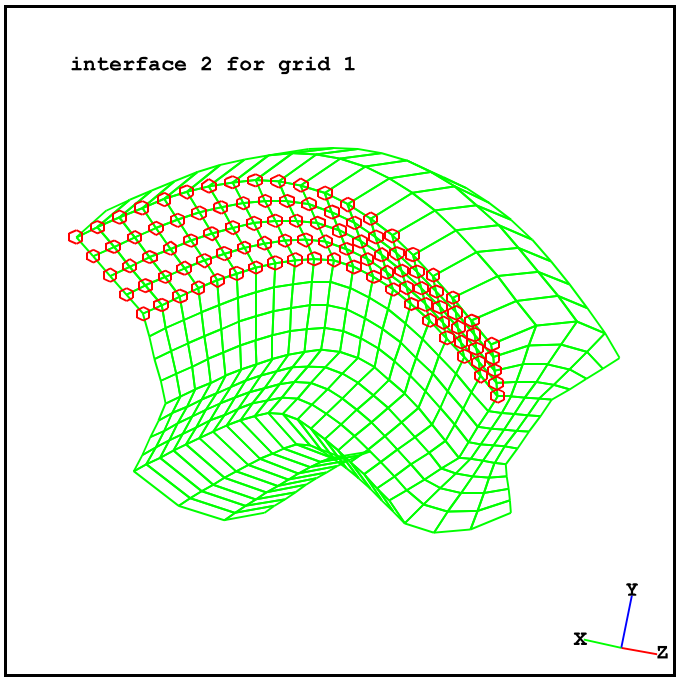


Figure 196 interface between grids

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
sd 1 plan 6 0 0 1 0 0
sd 2 plan 8 0 0 1 0 0
sd 3 plan 0 6 0 0 1 0
sd 4 plan 0 8 0 0 1 0
sd 5 cy 7 7 7 0 0 1 7
sd 6 sp 7 7 2 7
sd 7 sp 7 7 12 7
lcd 1 0 0 1 1;
block 1 6 7 9 10 15;
      1 6 7 9 10 15;
      1 6 10 14 18 23;
      5 5 6 8 9 9;
      5 5 6 8 9 9;
      -2 -2 2 12 16 16;
dei 1 2 0 5 6;1 2 0 5 6;;
dei 1 2 0 5 6;;1 2 0 5 6;
dei ;1 2 0 5 6;1 2 0 5 6;
dei 3 4;3 4;;
sfi -3;3 4;;sd 1
sfi -4;3 4;;sd 2
sfi 3 4;-3;;sd 3
sfi 3 4;-4;;sd 4
sfi -1 -6;-1 -6;3 4;sd 5
sfi -1 -6;-1 -6;-1 3;sd 6
sfi -1 -6;-1 -6;4 -6;sd 7
mom 3 3 1 4 4 1 1 1 x
endpart
merge
```

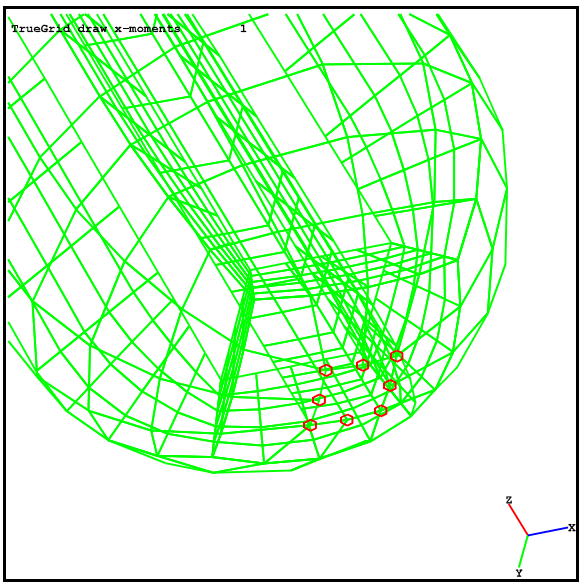


Figure 197 Wire drawing with moments

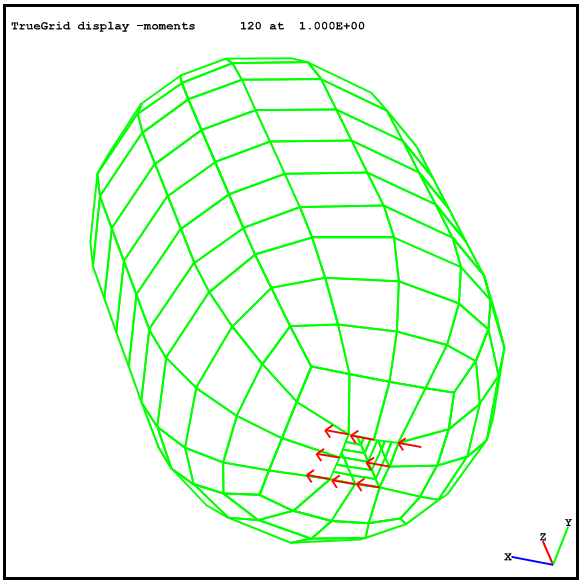


Figure 198 Hidden view with moments

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

This example takes two files. The second one is named row and is included in the first file.

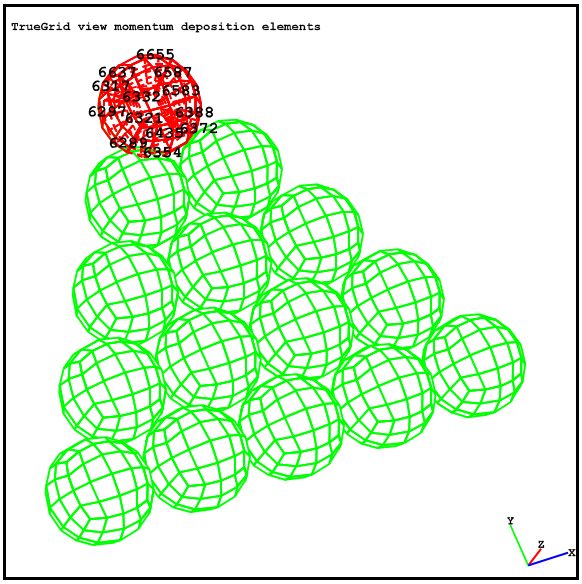


Figure 199 Parametric replication

```
sd 1 sp 0 0 0 3
para n 5
     x0 0
     y0 0;
include row
para n 4
     x0 3
     y0 [3*tan(60)];
include row
para n 3
     x0 [2*3]
     y0 [2*3*tan(60)];
include row
para n 2
     x0 [3*3]
     y0 [3*3*tan(60)];
include row
para n 1
     x0 [4*3]
     y0 [4*3*tan(60)];
include row
```

The second file, called row, contains the following commands:

```
block 1 5 9 13;1 5 9 13;1 5 9 13;
-1 -1 1 1 -1 -1 1 1 -1 -1 1 1
dei 1 2 0 3 4;1 2 0 3 4;;
dei 1 2 0 3 4;;1 2 0 3 4;
dei ;1 2 0 3 4;1 2 0 3 4;
sfi -1 -4;-1 -4;-1 -4;sd 1
if(%n.eq.1)then
 mdep 1 1 1 4 4 4 .2 2 0 .01
endif
gct 1 mx %x0 my %y0;grep 1;
lct %n mx 6;repe %n;lrep 1:%n;
endpart
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
accuracy 3

ld 1 lp2 0 0 1 1 2 4;;
     lp2 3 4 4 1 5 0;;

sd 1 crz 1
sd 2 plan 0 0 1 0 0 1
sd 3 plan 0 0 2.75 0 0 1
sd 4 cy 0 0 0 0 0 1 2.5
sd 5 plan 0 0 0 1 1 0
sd 6 plan 0 0 0 -1 1 0
sd 7 plan 0 0 0 0 0 1

block -1 -9 15 -21 -29;
      -1 -9 15 -21-29;
      1 5 -11;
      -5 -1 0 1 5
      -5 -1 0 1 5
      0 2 4
dei -2 -4;-2 -4;-1 -2;
dei -2 0 -4;1 2 0 4 5;;
dei 1 2 0 4 5;-2 0 -4;;
dei 2 4; 2 4; -3;
dei -1 2 0 4 -5;-1 2 0 4 -5;1 -3;
sfi -1 -2 -4 -5; -1 -2 -4 -5; 1 -3;sd 1
sfi ;; -2;sd 2
sfi -1 0 -5; 2 4; -3;sd 3
sfi 2 4; -1 0 -5; -3;sd 3
sfi -2 0 -4; 2 4; -3;sd 4
sfi 2 4; -2 0 -4; -3;sd 4
sfi -2; 4 5; 1 3;sd 5 sfi 1 2; -4; 1 3;sd 5
sfi -4; 1 2; 1 3;sd 5 sfi 4 5; -2; 1 3;sd 5
sfi -2; 1 2; 1 3;sd 6 sfi 1 2; -2; 1 3;sd 6
sfi -4; 4 5; 1 3;sd 6 sfi 4 5; -4; 1 3;sd 6
sfi ;; -1;sd 7 dei ; 1 3;;
plane 1 0 0 0 0 1 0 .001 symm ;
endpart
merge
grid on
```

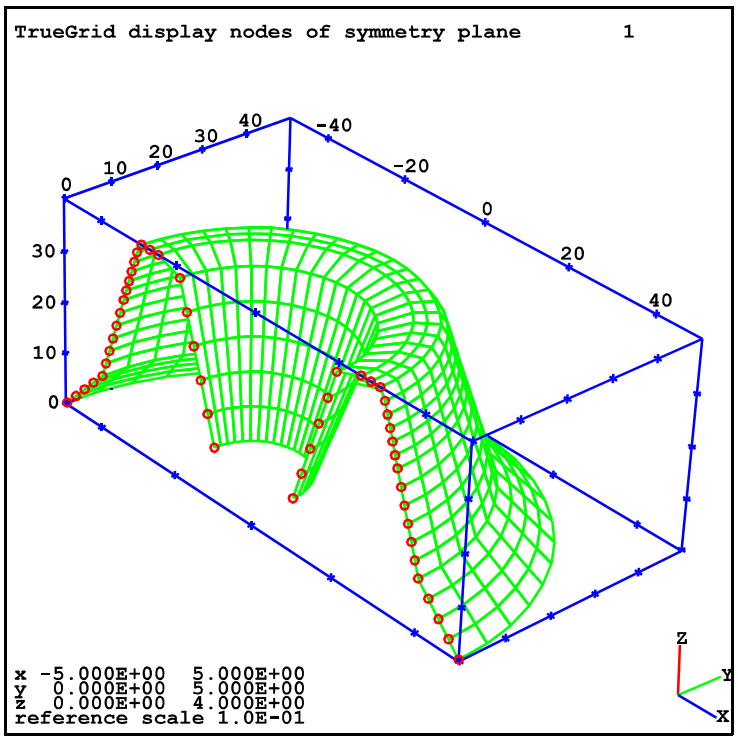


Figure 200 Boundary conditions from symmetry plane

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

co ol **display outlets**

Remarks

The **co il** command works identically to this command.

Example

```
sd 1 cyli 2.5 0 2.5 0 1 0 1
block 1 5 11 17 23 27;
      1 5 9;
      1 5 11 17 23 27;
      1 2 2 3 3 4;
      1 3 5;
      1 2 2 3 3 4;
dei 1 2 0 5 6;2 3;;
dei ; 2 3;1 2 0 5 6;
dei 2 3 0 4 5;2 3;2 3 0 4 5;
dei 2 3 0 4 5;;1 3 0 4 6;
dei 1 2 0 5 6;;2 3 0 4 5;
mbi -1 0 -6;;-4 0 -5;z .25
mbi -1 0 -6;;-2 0 -3;z -.25
mbi -2 0 -3;;-1 0 -6;x -.25
mbi -4 0 -5;;-1 0 -6;x .25
sfi -2 -5;;-2 -5;sd 1
bb 2 1 4 3 3 4 1;
bb 3 1 4 3 3 5 1;
bb 4 1 4 4 3 5 2;
bb 4 1 4 5 3 4 2;
bb 4 1 3 5 3 3 3;
bb 4 1 2 4 3 3 3;
bb 3 1 2 3 3 3 4;
bb 2 1 3 3 3 3 4;
esm 3 3 2 4 3 5 & 2 3 3 3 3 4 & 4 3 3 5 3 4 100 .00001 1 .6 4
esm 2 1 3 5 1 4 & 3 1 4 4 1 5 & 3 1 2 4 1 3 100 .00001 1 .6 4
unifm 3 1 2 4 3 5 & 2 1 3 3 3 4 & 4 1 3 5 3 4 200 .00001 1
il 1 3 1 6 3 6
ol 1 1 1 6 1 6
endpart
merge
```

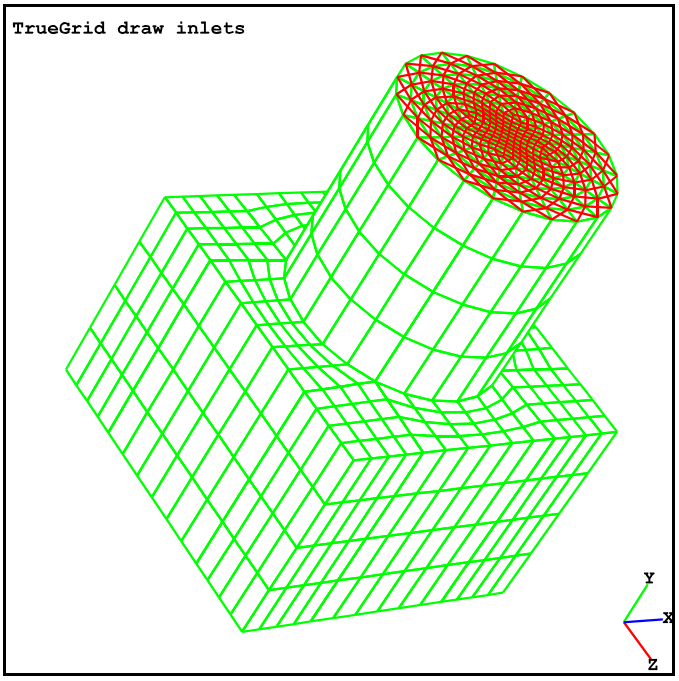


Figure 201 Inlet marked in hide mode

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

co spw **display numbered spot welds**

Example

```
sd 1 cy 1.1 1.1 0 0 0 1 1
lmi 1
block -1 6 21;-1 6;1 21;
      .1 1.1 5 .1 1.1 0 5
sfi -1 2; -1 2; 1 2;sd 1
y=y+.1*sin((k-1)*36)
lct 1 ryz; lrep 0 1;
mate 1
endpart
block -1;1 11 21;1 21;
       0 -2.9 1.1 5.1 0 5
dom 1 2 1 1 2 2
y=y+.1*sin((k-1)*36)
mate 3
endpart
merge
spw 21 spwd 1 node 672 node 1323
node 126 ;;
```

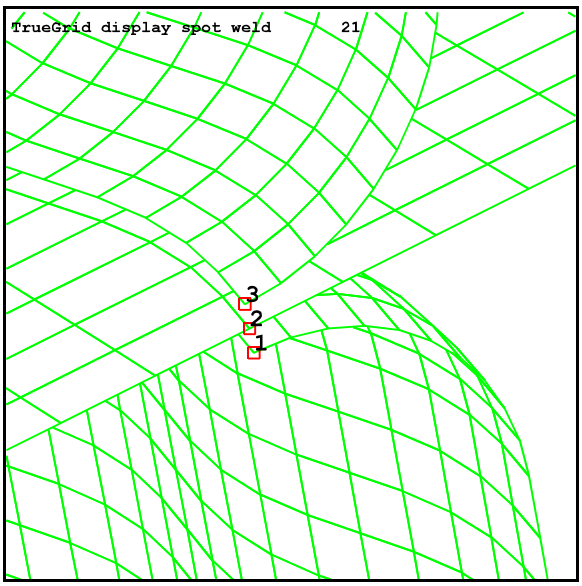


Figure 202 nodes of a numbered spotweld

co spwf **display facial spot welds**

Remarks

If the node defining the spot weld is not visible, a yellow box is draw.

Example

These commands are added to the previous example.

```
spwf rt -.1 1 4.875 3 mat 1 mat 2
spwf rt .1 .93 2.15 3 mat 1 mat 2
```

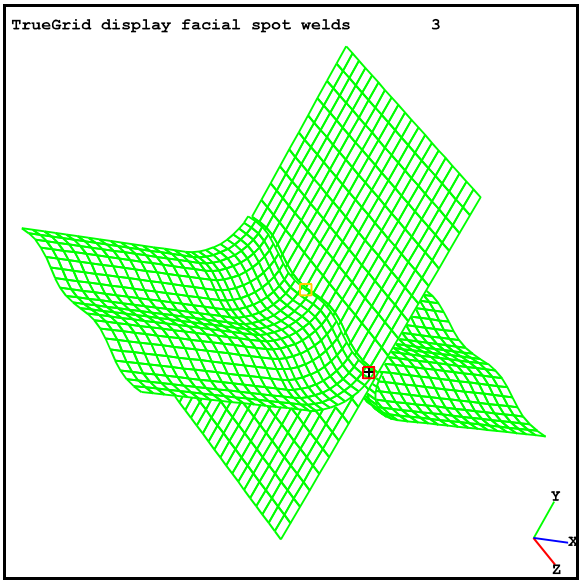


Figure 203 2 Facial spot welds

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

co jt **display a numbered joint**

Example

```
cylinder 1 3;1 13;1 6;
         1 2 0 360 0 5
cylinder 1 3;1 13;1 6;
         1 2 0 360 5.5 10.5
merge
bm n1 42 n2 6 n3 180
   mate 2 nbms 2 ;
bm n1 289 n2 253 n3 235
   mate 2 nbms 2 ;
jd 1 uj pnlt .33 ;
jt 1 1 n 470 ;
jt 1 2 n 469 ;
jt 1 3 n 253 ;
jt 1 4 n 42 ;
```

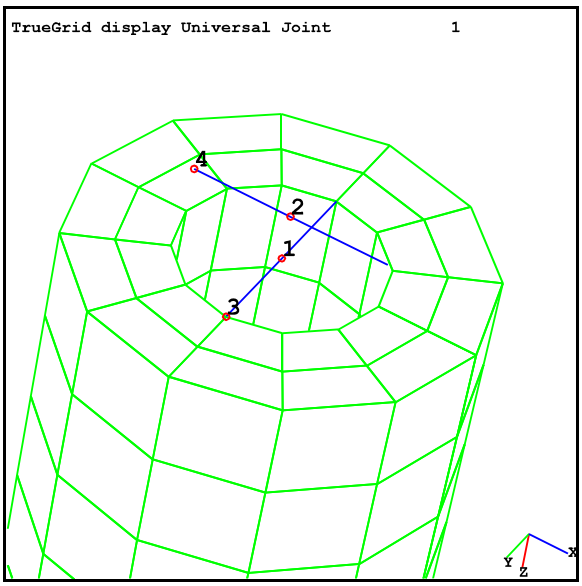


Figure 204 Nodes forming a joint

labels specify type of label to be displayed

labels *option*

or

la *option*

where *option* can be any of:

off	to turn off labels display
size *scale*	to scale the size of the tokens displaying labels
angle θ	to change the cone angle for the tokens displaying labels
nodes	to label nodes by node number
1d	to label 1D beam elements by element number
2d	to label 2D shell elements by element number
2q	to label 2D quadratic shell elements by element number
3d	to label 3D brick element faces by element number
3q	to label 3D quad'c brick element faces by element number
ijk1	for labeled and color coded reduced indices
ijk2	for labeled and color coded parts and reduced indices
ijk4	for color coded indices
locnd *node_number*	to locate a node by its number
loc1d *element_number*	to locate a 1D beam element by its number

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

loc2d *element_number*	to locate a 2D shell element by its number
loc3d *element_number*	to locate a 3D brick element by its number
loc2dq *element_number*	to locate a 2D quadratic shell element by its number
loc3dq *element_number*	to locate a 3D quadratic brick element by its number
sd	to display numbers of defined surfaces
crv	to display numbers of defined 3D curves
sdedge	to display surface edge identification numbers
sdpt	to display labels of points on defined surfaces
crvpt	to display labels of points on defined 3D curves
bb	to display the master block boundary interfaces
nodeset *set_name*	to display a node set
onset *set_name first last*	to display an ordered node set
faceset *set_name*	to display a face set
facesel *set_name*	to display and label a face set
elemset *set_name*	to display an element set
parts	to display parts
tol $part_1$ $part_2$	to display nodes merged between parts
where	
$part_1$	part sequence number (0 means all)
$part_2$	part sequence number (0 means all)
fraces	free faces (only actively displayed parts and materials)
fredges	free edges (only actively displayed parts and materials)
cracks θ	cracks below a specified subtended angle

Remarks

Labels do not overlap. If an object is not labeled, zoom in on it so there is more room for the label. Also try rotating the picture. The tokens used for a feature in the model may be different in the wire, hide, and fill graphics mode. These differences can used to your advantage. For example. when nodes are labeled in the wire mode, then nodes in the back of the mesh may also be labeled. Or, if the mesh is drawn in hide, only visible nodes are labeled. In another example, when using the fraces (free faces) option in wire mode, all free faces will be drawn in red and will stand out even in a large mesh. In hide mode, the hidden edges of free faces will appear as dotted lines. In fill graphics, they appear as red faces.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

labels nodes **label nodes by node number**

Example

Part 1 is composed of linear shells and Part 2 is composed of quadratic shells. The nodes are labeled in **Figure 205**. There are coincident nodes between parts, which are not merged.

```
linear
block 1 5;1 5;-1;1 5;1 5;0;
endpart
quadratic
block 1 5;1 5;-1;5 10;1 5;0;
merge
labels nodes
```

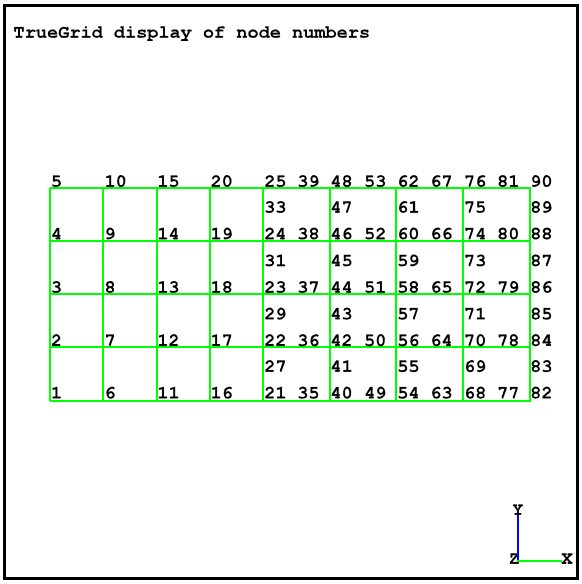


Figure 205 labeled nodes

labels 1d **label 1D beam elements by element number**

Example

20 beam elements are created along a curve. The elements are labeled. The simplified command file follows:

```
curd 1 arc3 seqnc rt -1 0 0 rt 0 1
0 rt 1 0 0 ;
curd 2 lp3 -1 -1 0 -1 0 0; ;
curd 3 lp3 1 -1 0 1 0 0; ;
curd 4 cpcds 2 1 3; ;
bm rt1 -1 -1 0 ; rt2 1 -1 0 ;
orient 0 0 0 mate 1 cs .1 nbms 20
cur 4 ;
labels 1d center
```

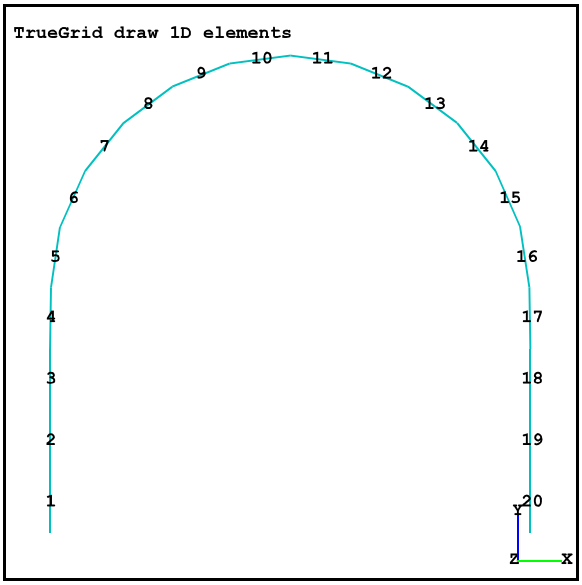


Figure 206 labeled beams

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

labels 2d **label 2D shell elements by element number**

Example

This example uses the mesh from the example for the **labels nodes** command. The numbering of elements always starts with 1 for each type of element.

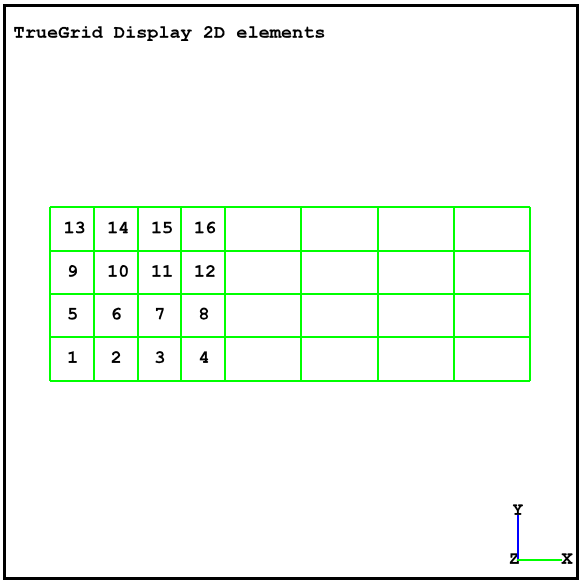


Figure 207 labeled linear shells

labels 2q **label 2D quadratic shell elements by element number**

Example

This example uses the mesh from the example for the **labels nodes** command. The numbering of elements always starts with 1 for each type of element.

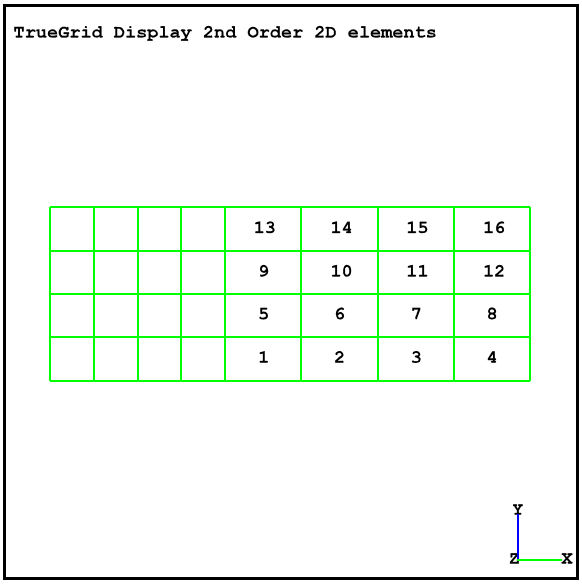


Figure 208 labeled quadratic shells

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

labels 3d **label 3D brick faces by element number**

Example

Part 1 is composed of linear bricks and Part 2 is composed of quadratic bricks. The numbering of elements always starts with 1 for each type of element. The linear bricks are labeled by the element numbers. The simplified command file follows:

```
linear
block 1 5;1 5;1 5;1 5;1 5;1 5;
endpart
quadratic
block 1 5;1 5;1 5;5 12;1 5;1 5;
merge
labels 3d
```

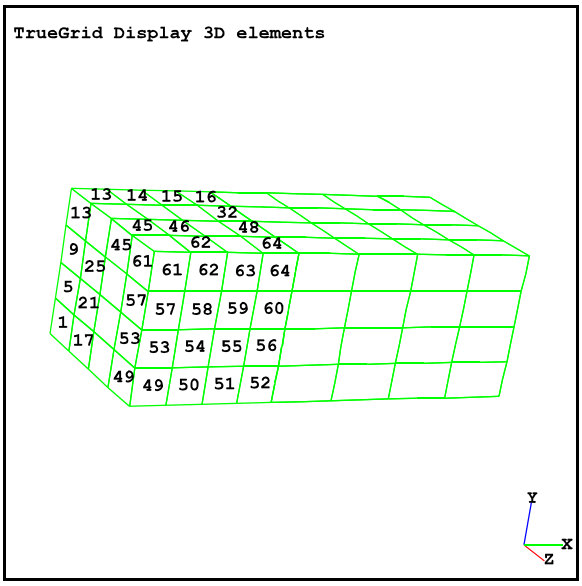


Figure 209 labeled bricks

labels 3q **label 3D quadratic brick faces by element number**

Example

This example uses the mesh from the example for the **labels 3d** command. The numbering of elements always starts with 1 for each type of element. The quadratic bricks are labeled by the element numbers.

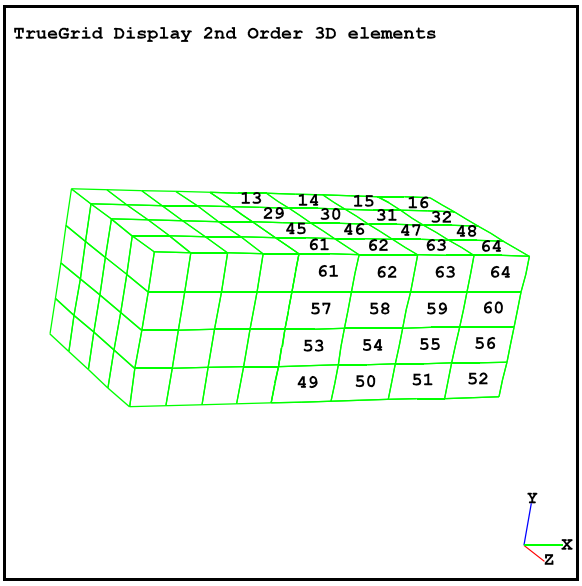


Figure 210 labeled quadratic bricks

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

labels ijk1 **display color coded reduced indices**

Remarks

labels ijk1 can be used after issuing the Hide option in the Merge Phase. (I.e., this works when looking at a hidden line drawing.)

You can view the part information in the Merge Phase in a similar manner as in the Part Phase.

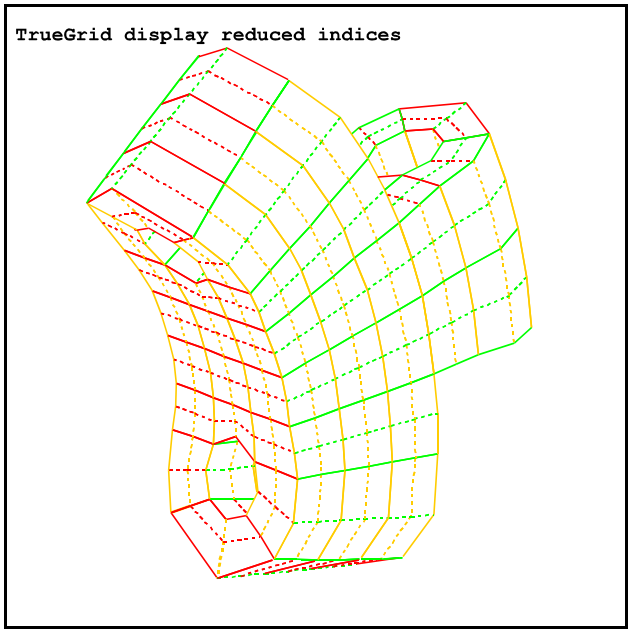


Figure 211 color coded reduced indices

labels ijk2 **display color coded and numbered reduced indices**

Remarks

Labels ijk2 can be used after issuing the Hide option in the Merge Phase.

You can view the part information in the Merge Phase in a similar manner as in the Part Phase.

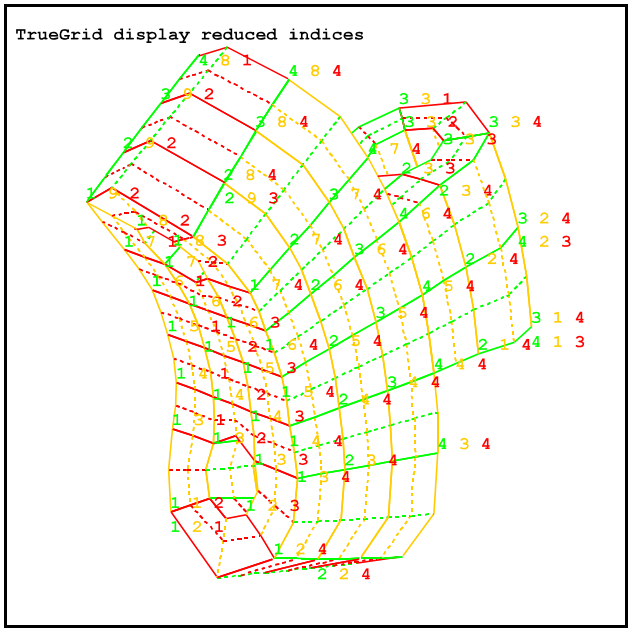


Figure 212 numbered reduced indices

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

labels ijk4 **display color coded/numbered reduced indices/parts**

Remarks

labels ijk4 can be used after issuing the Hide option in the Merge Phase.

You can view the part information in the Merge Phase in a similar manner as in the Part Phase.

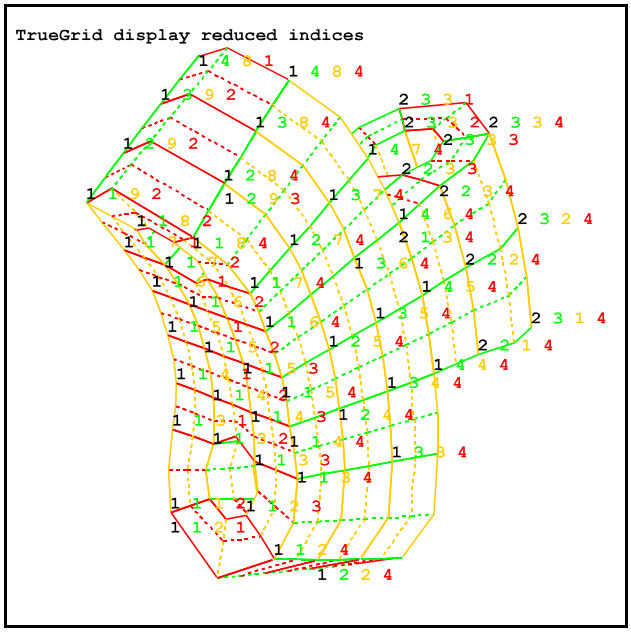


Figure 213 reduced indices and parts

labels locnd **locate a node by its number**

locnd *node_number*

Example

This example uses the mesh from the example for the **labels nodes** command. The node 20 is located. The located node is in the center of the picture by default.

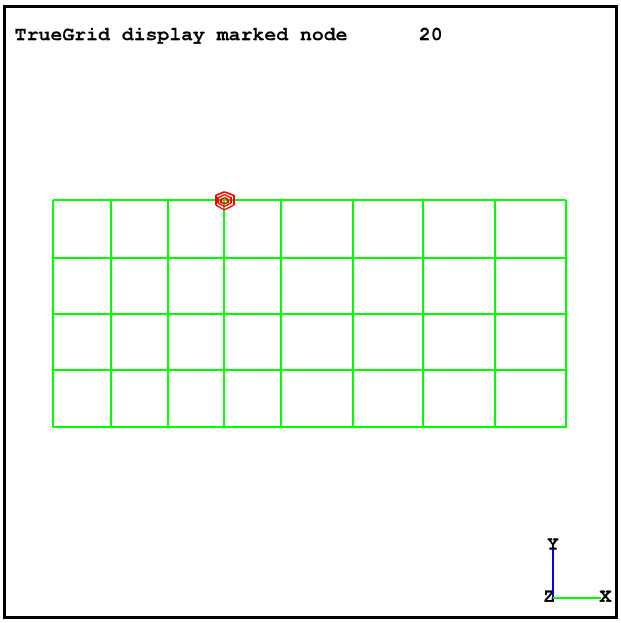


Figure 214 located node

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

labels loc1d **locate a 1D beam element by its number**

loc1d *element_number*

Example

This example uses the mesh from the example for the **labels 1d** command. The beam 5 is located. The located beam is in the center of the picture by default.

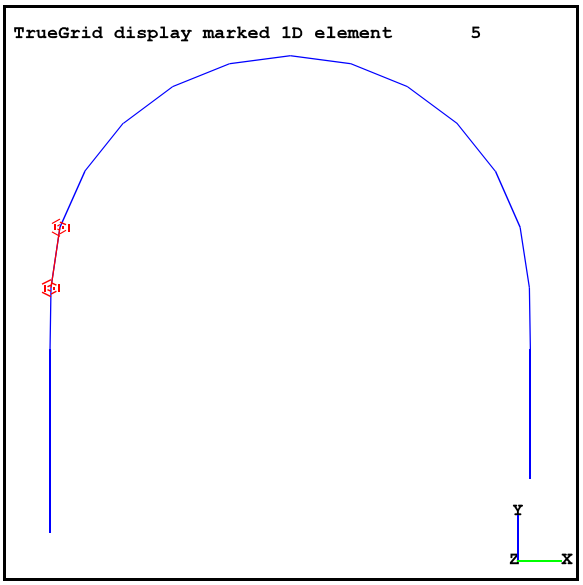


Figure 215 located beam

labels loc2d **locate a 2D shell element by its number**

loc2d *element_number*

Remarks

The diagonals of a shell are displayed to help check distorted elements.

Example

This example uses the mesh from the example for the **labels 2d** command. The shell number 10 is located. The located shell is in the center of the picture by default.

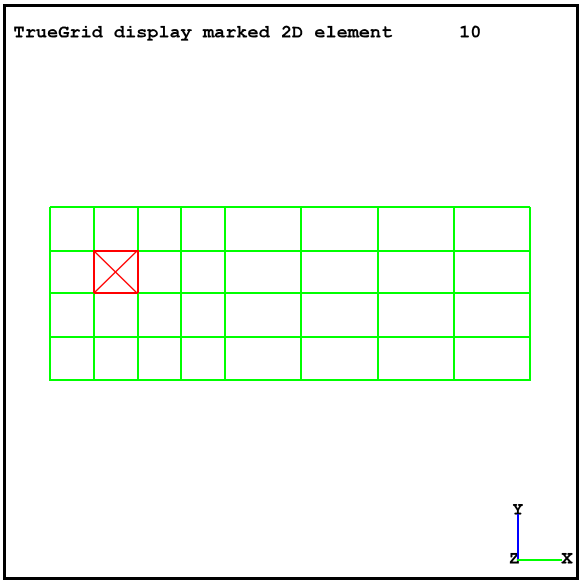


Figure 216 located linear shell

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

labels loc2dq **locate a 2D quadratic shell element by its number**

loc2dq *element_number*

Remarks

The diagonals of a shell are displayed to help check distorted elements.

Example

This example uses the mesh from the example for the **labels 2d** command. The quadratic shell number 10 is located. The located shell is in the center of the picture by default.

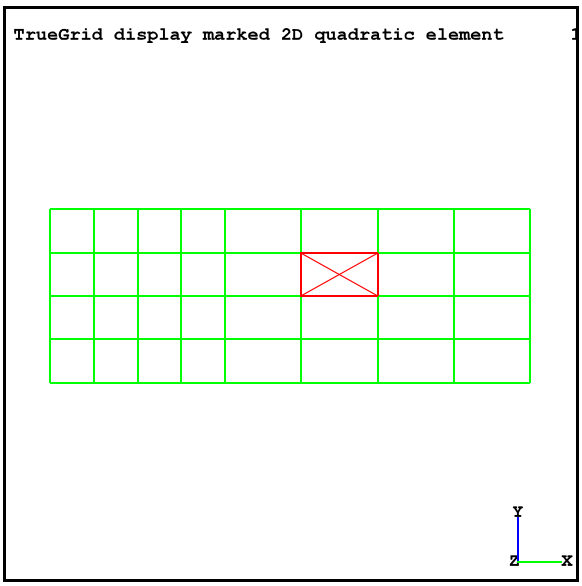


Figure 217 located quadratic shell

labels loc3d **locate a 3D brick element by its number**

loc3d *element_number*

Remarks

The diagonals of brick faces are displayed to help check distorted elements.

Example

This example uses the mesh from the example for the **labels 3d** command. The brick number 10 is located. The located brick is in the center of the picture by default.

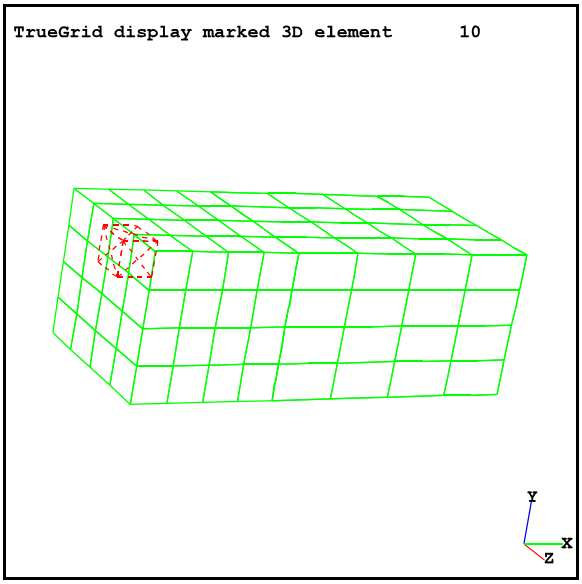


Figure 218 located brick

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

labels loc3dq **locate a 3D quadratic brick element by its number**

loc3dq *element_number*

Remarks

The diagonals of brick faces are displayed to help check distorted elements.

Example

This example uses the mesh from the example for the **labels 3d** command. The quadratic brick number 10 is located. The located brick is in the center of the picture by default.

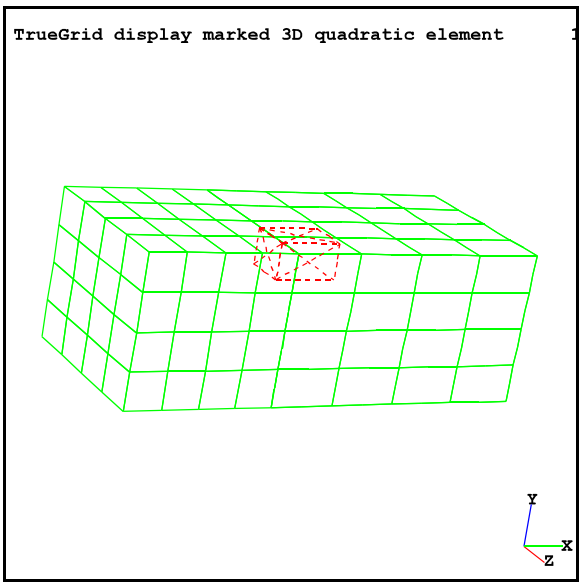


Figure 219 located quadratic brick

labels sd **label surfaces**

Example

```
curd 1 arc3 seqnc rt -1 0 0 rt 0 1
0 rt 1 0 0 mz -2;
curd 2 arc3 seqnc rt -1 0 0 rt 0 1
0 rt 1 0 0 mz 2;
curd 3 lp3 -1 0 -2 -1 0 2;;
curd 4 lp3 1 0 -2 1 0 2;;
curd 5 lp3 2 0 -2 2 0 2;;
curd 6 lp3 -3 0 -2 -3 0 2;;
curd 7 lp3 -3 -2 -2 -3 -2 2;;
curd 8 lp3 1 0 -2 2 0 -2;;
curd 9 lp3 1 0 2 2 0 2;;
curd 10 lp3 -1 0 2 -3 0 2;;
curd 11 lp3 -1 0 -2 -3 0 -2;;
curd 12 lp3 -3 0 -2 -3 -2 -2;;
curd 13 lp3 -3 0 2 -3 -2 2;;
sd 1 blend4 1 3 2 4;
sd 2 blend4 9 5 8 4;
sd 3 blend4 10 3 11 6; sd 4 blend4
13 7 12 6;dasd labels sd
```

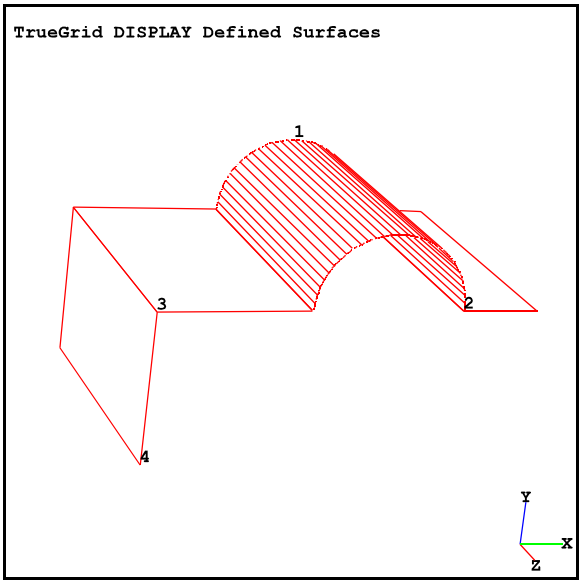


Figure 220 labeled surfaces

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

labels crv **label curves**

Remarks

The **labels crv** command labels curves. It displays labels of curves in a non-overlapping way. To display more details use the **zf** or **zb** commands (zoom forward, zoom back).

Example

Curves and surfaces are defined. The curves are labeled in .

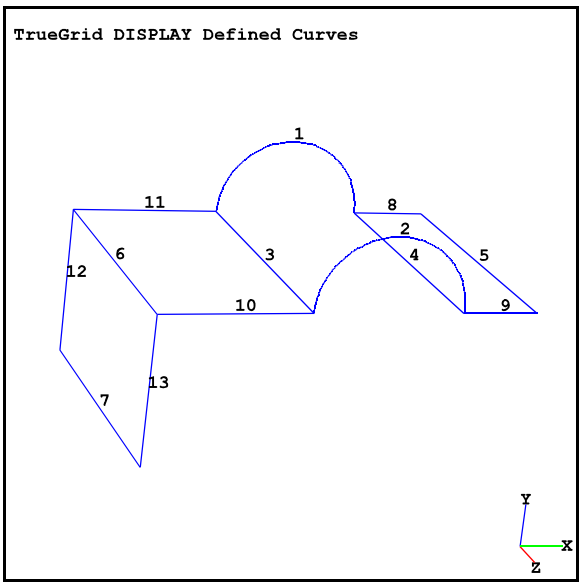


Figure 221 labeled curves

labels sdedge **label surface edges**

Remarks

This command's functionality can be reproduced by using the ***Labels*** button in the Environment Window. The **labels sdedge** command labels edges of surfaces. It displays labels of surface edges in a non-overlapping way. To display more details use **zf** or **zb** command (zoom).

The label has the notation: *surface-#.edge-#*.

Example

Curves and surfaces are defined. The surface edges are labeled in 222.

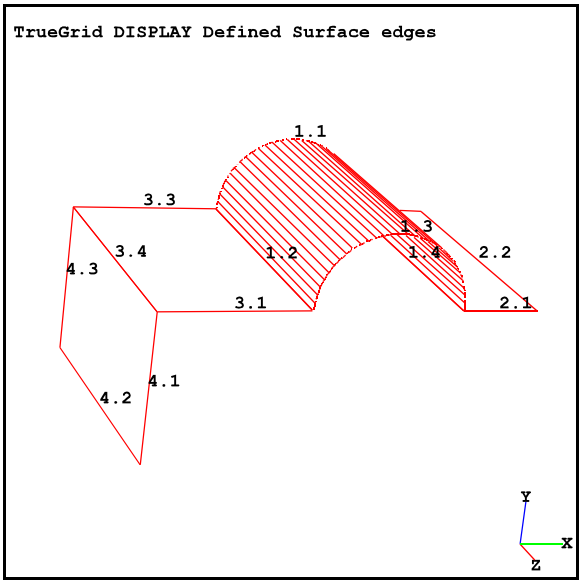


Figure 222 surface edges are labeled

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

labels sdpt **label surface points**

Remarks

The **labels sdpt** command labels surface definition points. It displays labels of surface definition points in a non-overlapping way. To display more details use **zf** or **zb** command (zoom).

The label has notation *surface_#.i_contour_#.j_contour_#* for most surfaces except for triangles from the IGES, VPOINT and STL files. For triangles the notation is *surface_#.triangle_#.point_#*, where *point_#* can be 1, 2 or 3.

See pg.? for a discussion on tessellation.

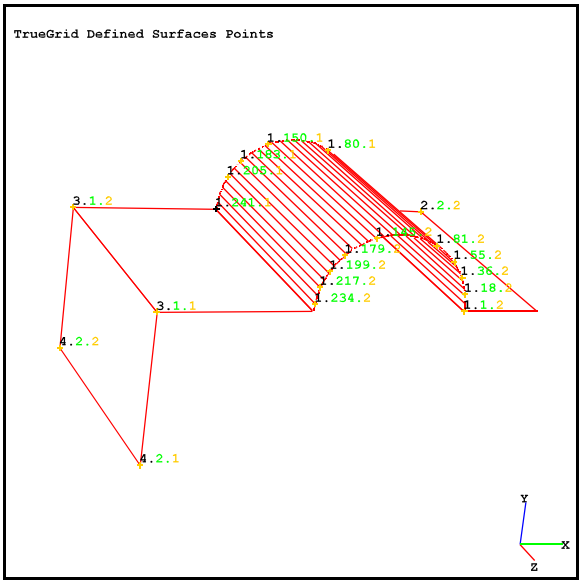


Figure 223 labeled surface definition points

labels crvpt **label curve points**

Remarks

The **labels crvpt** command labels curve definition points. It displays labels of curve definition points in a non-overlapping way. To display more details use the **zf** or **zb** commands (zoom).

The label has notation *curve.i.j* .

Example

Curves and surfaces are defined. The curve definition points are labeled in **Figure 224**.

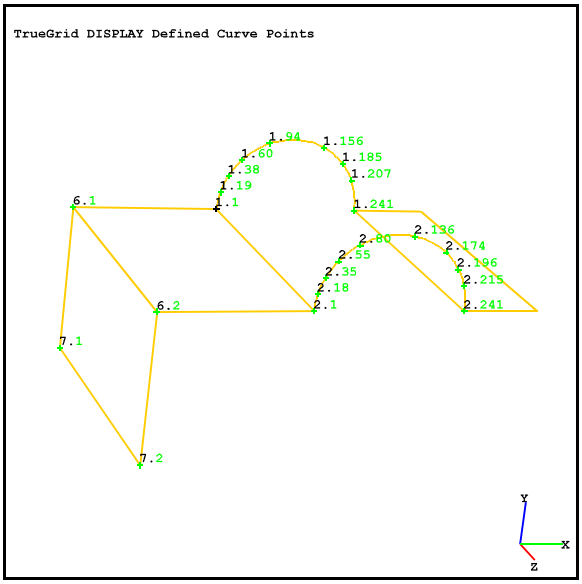


Figure 224 labeled curve definition points

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

labels faceset **display face set**

Remarks

labels faceset command works similarly as the **labels facesel** command, but without labeling.

labels facesel **display and label face set**

The faces in the face sets are labeled in a consistent fashion:

b-*number_of_element.number_of_face*	brick element
qb-*number_of_element.number_of_face*	quadratic brick element
s-*number_of_element.number_of_face*	shell element
qs-*number_of_element.number_of_face*	quadratic shell element

where

number_of_element	is the number of the element
number_of_face	is the number of the face (1-6)

Remarks

See also pg. ? for the face sets definition in the Part Phase.

Example

The face set **s1** is initialized by region, then it is enhanced 3 times by regions from various parts. The elements in the parts are linear bricks, quadratic bricks, linear shells and quadratic shells. The face set **s1** is displayed and labeled in **Figure 225**. The simplified command file follows:

```
linear          c set the type of element to linear

c mesh definition - linear bricks
block 1 3 5 7;1 3 5 7;1 3;1 2 3 4 1 2 3 4 -1 1

fset 1 1 2 4 4 2 = s1          c face set initialization
```

... mesh manipulations ...

```
c mesh definition - linear shells
block 1 3 5 7;1 3 5 7;-1;1 2 3 4 1 2 3 4 1
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
fset 1 1 1 4 4 1 u s1          c face set union
```

... mesh manipulations ...

```
endpart

quadratic      c set the type of element to quadratic

c mesh definition - quadratic bricks
block 1 3 5 7;1 3 5 7;1 3;1 2 3 4 1 2 3 4 -1 1

fset 1 1 2 4 4 2 u s1          c face set union
```

... mesh manipulations ...

```
c mesh definition for quadratic shells
block 1 3 5 7;1 3 5 7;-1;1 2 3 4 1 2 3 4 1

fset 1 1 1 4 4 1 u s1          c face set union
```

... mesh manipulations ...

```
merge
labels facesel s1
```

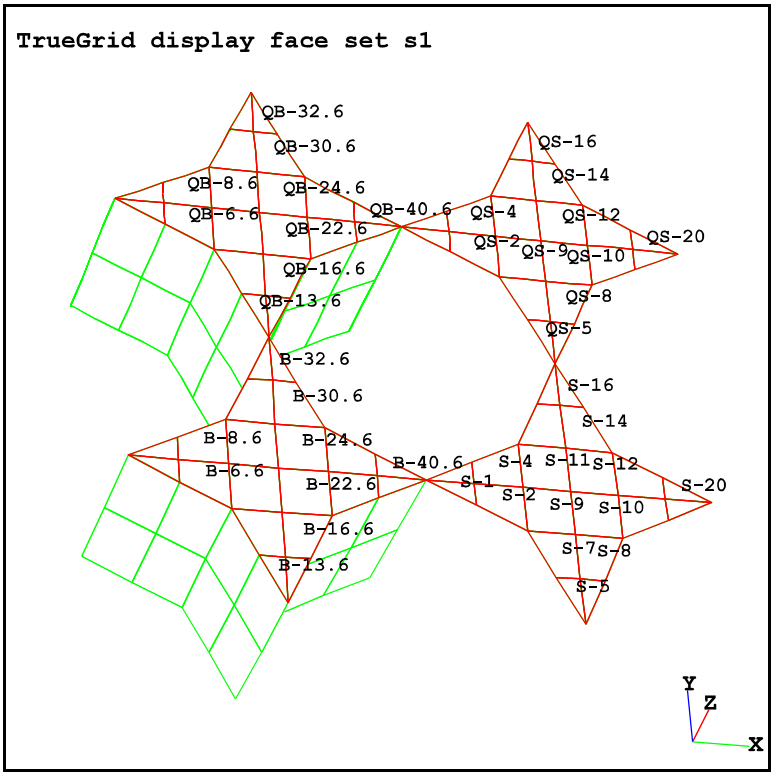


Figure 225 results of the **facesel** command

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

labels parts **label parts**

Remarks

Clicking the left mouse button on the part label displays a skeleton of the part.

Example

Two parts are created and labeled.

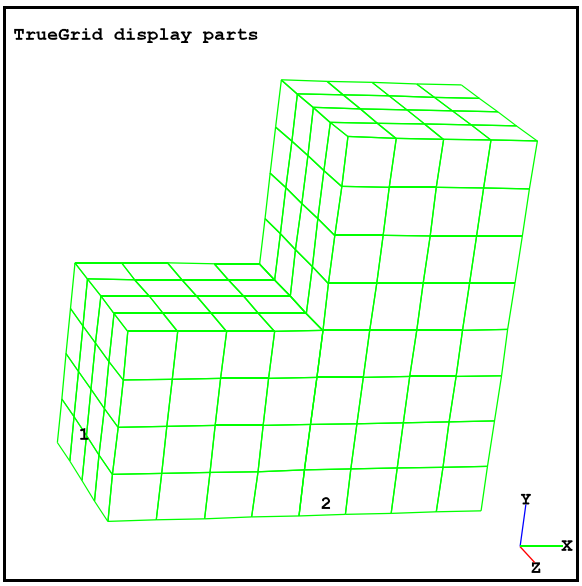


Figure 226 labeled parts

labels tol **display nodes merged between parts**

tol *part*$_1$ *part*$_2$

where

part$_1$	part number (0 means all)
part$_2$	part number (0 means all)

See pg. 201, 434 for the rules which govern the merging of the nodes.

Example

Two parts are created. The common nodes are merged and displayed. The simplified command file follows:

```
stp .001 c merge nodes
labels tol 1 2
```

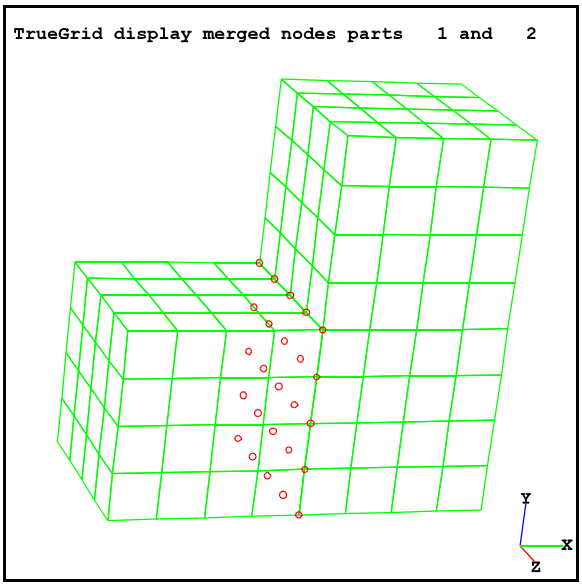


Figure 227 merged nodes

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

labels fraces **display free faces**

By definition, at least one node is not merged on the free face. Free faces are constructed from the faces of the brick elements. Free faces are displayed only for active parts and materials in the picture. It is advantageous to label free faces only for a small number of parts. See pg.201, 434 for merging of the nodes. After invoking this option, you can remove or add parts and materials and the red faces will remain in the picture.

Example

The mesh from the **labels parts** example is used. The common nodes are merged. Free faces on the Part 2 are displayed.

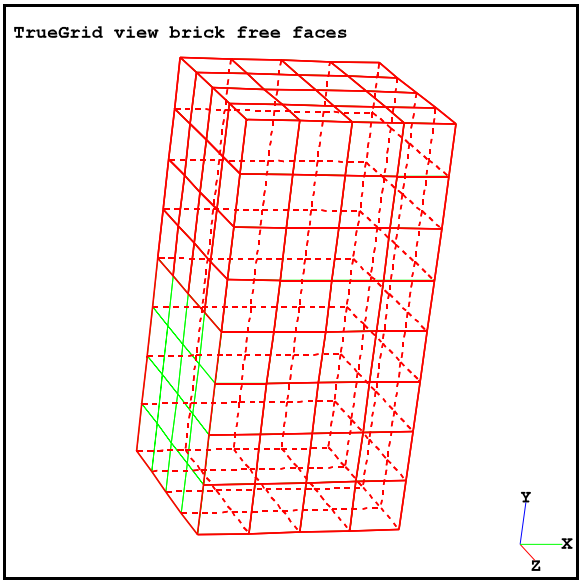


Figure 228 displayed free faces on Part 2

labels fredges **display free edges**

Remarks

By definition, at least one node is not merged on a free edge. Free edges are constructed from the edges of the shell elements. Free edges are displayed only for active parts in the picture. It is advantageous to label free edges only for a small number of parts. See pg. 201, 434 for the rules which govern the merging of the nodes. Only parts in the picture are considered.

Example

The mesh is created. The common nodes are merged. Parts 7 8 9 10 11 26 are displayed. Free edges are displayed.

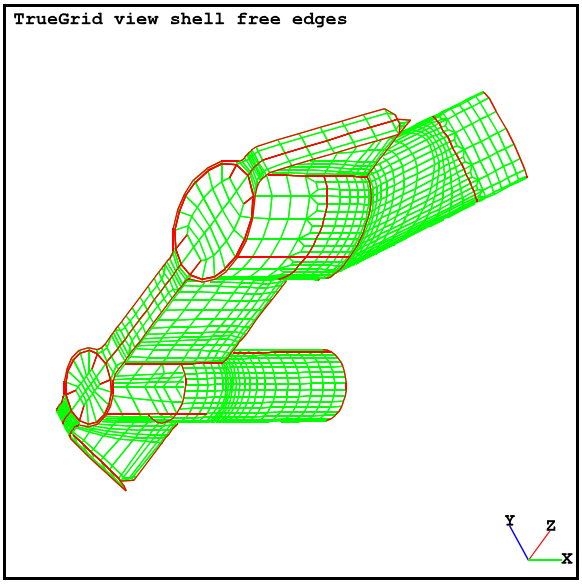


Figure 229 displayed free edges on shells

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

labels cracks θ **display cracks in the mesh**

Remarks

The angle is used to measure between the exterior polygons. If the angle is below the specified angle, it is labeled as a crack. This is many times the minimum data needed to detect a problem in the node merging.

Example

The butterfly mesh has not been merged, so this algorithm finds the critical spot where the merging first failed.

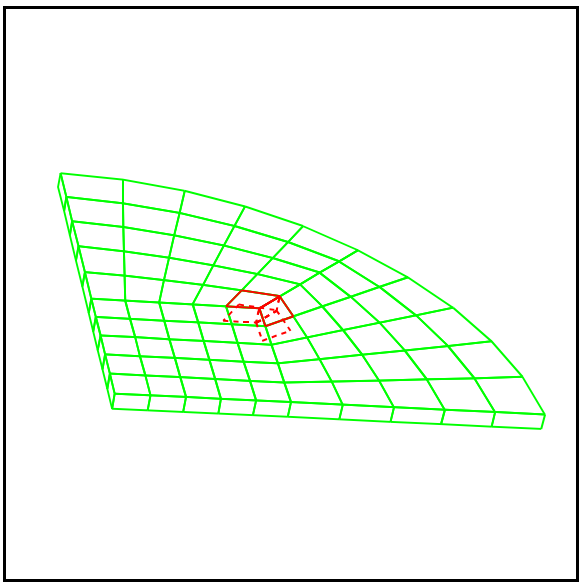


Figure 230 Crack in the mesh

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

5. Animation Commands

You define a set of images and then generate additional images between these keyframe images in order to provide a smooth animation. This can give the appearance of flying around and even through a model.

To create an animation, you first establish up a few keyframe pictures. You to establish up to 30 keyframes. There is no limit to the number of interpolated images that may be generated.

The **sv** command will save a keyframe or view. You can go back to look at a single picture with **shv**. These two commands alone are useful for demonstrations and debugging. But with **avc** or **av** you can get an animated view of your model. The interpolated pictures allow you to produce a smooth animation while having to save a minimal number of images by hand.

With the **postscript** command you can save all these pictures as PostScript files for more processing.

av animate views - linear interpolation

av *graphics #views* $view_1$ $\#frames_{12}$ $view_2$ $\#frames_{23}$ ...

where

graphics can be

poor	for a wireframe image,
disp	for a hidden line image, or
tvv	for a fill mode image

view1, *view2*,...	are the view numbers of views saved with **sv**, and
$\#frames_{ij}$	is the number of frames to interpolate between views numbered $view_i$ and $view_j$.

Remarks

Linear interpolation is used to form the specified number of intermediate pictures. This means that all of the parameters controlling the orientation of the picture are interpolated equally to produce a constant change in the orientation from one frame to another.

You can sequence up to 30 saved views. There is no limit on the number of frames interpolated between them.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

avc animate views - cosine interpolation

avc *graphics #views view$_1$ #frames view$_2$ #frames ...*

where

graphics can be

poor	for a wireframe image,
disp	for a hidden line image, or
tvv	for a fill mode image

view1, *view2*,... are the view numbers of views saved with **sv**, and

#frames$_{ij}$ is the number of frames to interpolate between views numbered *view$_i$* and *view$_j$*.

Remarks

Cosine interpolation is used to form the specified number of intermediate pictures. This means that all of the parameters controlling the orientation of the picture are interpolated in a manner that gives more frames near the saved views than midway between saved views.

You can sequence up to 30 saved keyframess. There is no limit on the number of frames interpolated between them.

shv show a saved view

shv *view_number*

where *view_number* is the number of a view saved with the **sv** command.

Remarks

This feature restores the picture orientation to a previously saved orientation. This can be useful for demonstrations, debugging, or designing an animation.

sv save view

sv *view_number*

where *view_number* is a unique number to identify the view

Remarks

You can save up to 30 keyframes, or views. You can then use these views in the other commands in this section.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

6. Exploded Views

These commands give you exploded views of your model. They affect only the graphics, not the actual model.

To make an exploded view of your model, you issue the **pexp** command to move some or all of the parts, with options of moving some parts farther than others, or in different directions. You can also issue the **mexp** command to move some or all of the materials, with options of moving some materials farther than others, or in different directions. The affect of these two commands issued multiple times is cumulative.

The following two explosion commands made the exploded view of the spaceship in the Introduction.

```
pexp 0 0 2   3 ;;
pexp 0 2 0   2 6; 1 5 ;;
```

The first command moved part 3 (front of secondary hull) to the right. The second command moved parts 2 and 6 up two units. Part 2 is the truss connected to the primary hull, and part 6 is the power unit supports - there are two supports modeled with one part. The second command also moved parts 1 and 5 up four units. Part 1 is the primary hull and part 5 is the power units - there are two power units modeled with one part. Part 4, the secondary hull, was not moved.

exp reactivate exploded views

Display an exploded view that you had previously turned off with an **expoff** command.

exp (no arguments)

Remarks

The exploded view will be based on the translations you specified in the **pexp** and **mexp** commands.

expoff turn off exploded views

expoff (no arguments)

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

iniexp initialize explode offset to zero

Restore every part and material to an offset of zero.

iniexp (no argument)

mexp offset each subset of materials past the previous subset

mexp *x y z list_of_material_# ; list_of_material_# ; ... ;*
where (*x*,*y*,*z*) is the first vector offset.

Remarks

Elements with material numbers are in the N-th list will be offset in the picture by N times the specified vector.

There may be up to 50 different lists of material numbers.

If you issue both a **mexp** and a **pexp** command, the offsets you specify will be added to each other.

For a 3-D exploded view, use more than one **mexp** or **pexp** command to translate different sets of parts in different directions.

pexp offset each subset of parts past the previous subset

pexp *x y z list_of_part_# ; list_of_part_# ; ... ;*

Remarks

Parts in the N-th list will be offset in the picture by N times the specified vector.

There may be up to 50 different lists of part numbers.

If you issue both a **mexp** and a **pexp** command, the offsets you specify will be simply added to each other.

For a 3-D exploded view, use more than one **mexp** or **pexp** command to translate different sets of parts in different directions.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

sclexp explode scale factor

Multiply all the offsets for exploded views by a single scale factor.

sclexp *scale_factor*

7. Material Commands

tmm specify the total mass of a material

tmm *material_# total_mas*s

Remarks

When the Merge Phase begins or when this command is issued in the Merge Phase, then the volumes of the specified materials are calculated. The density of the material is determined in order to satisfy the total mass requirement. This feature is only available for the D Y N A 3 D , N I K E 3 D , NASTRAN, TOPAZ3D, ES3D, ALE3D, ABAQUS, ANSYS, and NEUTRAL output options. One of these options must be selected before density can be determined. This feature applies to linear shells, bricks, and beams. The shell thickness and beam thickness must be set for shells and beams respectively.

buoy specify a buoyancy condition for a list of materials

buoy *material_list ; options ;*
where
options can be
bform *type*
where *type* can be
1 displaced volume method
2 distributed pressure (head) method (brick, shell only)
styp *type parameters*
where *type* and *parameters* can be
1 *xpt ypt zpt xvec yvec zvec* point on surface, downward vector
2 *node1 node2* node1 on surface, node2 directly beneath
lcid *load_id*
where
load_id = 0 specifies a stationary surface

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

load_id > 0 specifies the load curve for surface motion vs. time

rhofg *density*

where

density > 0 means a constant density

density < 0 is the negative of a load curve for density vs. time

mgth *thickness*

rhomg *density*

where

density > 0 means a constant density

density < 0 is the negative of a load curve for density vs. time

snid *type parameters*

where *type* and *parameters* can be

1	RH rule applied to local node numbering
2 *xpt ypt zpt*	toward a point P
3 *xyt ypt zpt*	away from a point P
4 *node*	toward node N
5 *node*	away from node N

broach *flag*

where

flag can be

1	neglect beam thickness as beam broaches surface
2	include beam thickness as beam broaches surface

am add material to the picture

am *material_number*

Remarks

The specified material is added to the list of materials to be displayed.

ams add a list of materials to the picture

ams *material_list ;*

Remarks

The specified list of materials are added to the list of materials to be displayed.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

dam display all materials in the picture

dam *(no arguments)*

Remarks

All materials are added to the materials display list.

dms display a set of materials in the picture

dms *material_list* ;

Remarks

The specified list of materials becomes the list of materials to be displayed.

dm display one material in the picture

dm *material_number*

Remarks

The material display list is set to the specified material.

ram remove all materials from the picture

ram

Remarks

No materials displayed.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

rm remove one material from the picture

rm *material_number*

Remarks

The specified material is removed from the list of materials to be displayed.

rms remove a list of materials from the picture

rms *material_list ;*

Remarks

The specified materials are removed from the list of materials to be displayed.

8. Interface Commands

iss save interface segments

iss fset *face_set*

Interface element commands (sliding lines and surfaces, etc.) that are available during the Merge Phase.

si select nodes for the slave side of a nodal sliding interface

si *type interface_# boundary options ;*

where

type can be one of:

n *node_numbe*r	to select a single node
rt *x y z*	to select a node close to a Cartesian point
cy *rho theta z*	to select a node close to a cylindrical point
sp *rho theta phi*	to select a node close to a spherical point
nset *name_of_set*	to select an entire node set
fset *face_set*	to select a face set

boundary can be one of

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

m master side of the interface
s slave side of the interface

options can be

fail *norm_failure_force shear_failure_force*
exp *norm_failure_exp shear_failure_exp*
fsf *coulomb_friction_scale viscous_friction_scale*

Remarks

The global properties of a sliding interface are defined using the **sid** command. Some interface types allow for nodes on the slave side. Most require face sets for both the master and slave sides. The dummy sliding interface type, which is used to control the merging without the side effect of causing a sliding interface definition in the output, allows for nodes on both the master and slave side.

9. Springs, Dampers, and Point Masses

npm creates a new node and assigns a point mass to it

npm *mp_node_# x y z mass options ;*

where

an *option* can be:

dx	no nodal displacement in the x-direction
dy	no nodal displacement in the y-direction
dz	no nodal displacement in the z-direction
rx	no nodal rotations about the x-axis
ry	no nodal rotations about the y-axis
rz	no nodal rotations about the z-axis
mdx	no mass displacement in the x-direction
mdy	no mass displacement in the y-direction
mdz	no mass displacement in the z-direction
mrx	no mass rotations about the x-axis
mry	no mass rotations about the y-axis
mrz	no mass rotations about the z-axis
ixx *mom*	specify the moment of inertia about the x-axis
iyy *mom*	specify the moment of inertia about the y-axis
izz *mom*	specify the moment of inertia about the z-axis
pdamp *alpha*	proportional damping factor (ABAQUS)
cdamp *fraction*	fraction of critical damping (ABAQUS)

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

This newly created node is separate from the existing mesh and can be attached by generating a beam or spring using this new node (see **bm** or **spring**). It can also be attached to the rest of the mesh by merging it to a neighboring node (see **t**, **tp**, **stp**, **bptol**, and **ptol)**. This is distinguished from assigning a mass to an existing node of the mesh. The latter can be done using the **pm** command. To create a new node and assign it a point mass such that it is replicated or transformed along with the part, then use the **npm** command in the Part Phase (see **lrep**, **grep**, and **pslv**). To assign a point mass to a vertex of a part such that it is replicated or transformed along with the part, use the **pm** command in the Part Phase. All of the options are not needed by all output options.

pm assigns a point mass to a node of the mesh

pm *node_# mass otions ;*

where

an *option* can be:

mdx	no mass displacement in the x-direction
mdy	no mass displacement in the y-direction
mdz	no mass displacement in the z-direction
mrx	no mass rotations about the x-axis
mry	no mass rotations about the y-axis
mrz	no mass rotations about the z-axis
ixx *mom*	specify the moment of inertia about the x-axis
iyy *mom*	specify the moment of inertia about the y-axis
izz *mom*	specify the moment of inertia about the z-axis
pdamp *alpha*	proportional damping factor (ABAQUS)
cdamp *fraction*	fraction of critical damping (ABAQUS)

Remarks

This is distinguished from creating a new node separate from the mesh and assigning a mass to it. The latter can be done using the **npm** command. To assign a point mass to a vertex within a part such that it is replicated or transformed along with the part, use the **pm** command in the Part Phase (see **lrep**, **grep**, and **pslv**). In order to create a new node and assign it a point mass such that it is replicated or transformed along with a part, then use the **npm** command in the Part Phase. All of the options are not needed by all output options.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

pminfo table of point masses information

pminfo (no arguments)

Example

The following example of the **pminfo** command was preceded by two point mass commands, the first (**npm** command) found in a part which was duplicated once, and the second (**pm** command) found in the Merge Phase.

```
npm 2 0 0 0 1.2 inc 2 dx rx mdx mrx iyy .1 izz .2 pdamp 2.3
    cdamp 1.2;
pm 377 2.1 mdx ixx .2 iyy .4 mrz pdamp .3 cdamp 2.3 ;

   STRUCTURED POINT MASSES TABLE
    node=     377, mass= 2.100E+00
     xxi= 2.000E-01, yyi= 4.000E-01, zzi= 0.000E+00
     proportional damping= 3.000E-01, critical damping= 2.300E+00
     dx=1, dy=0, dz=0, rx=0, ry=0, rz=1
   UNSTRUCTURED POINT MASSES TABLE
    number=   2 node=     126, mass= 1.200E+00
     xxi= 0.000E+00, yyi= 1.000E-01, zzi= 2.000E-01
     proportional damping= 2.300E+00, critical damping= 1.200E+00
     dx=1, dy=0, dz=0, rx=1, ry=0, rz=0
    number=   4 node=     252, mass= 1.200E+00
     xxi= 0.000E+00, yyi= 1.000E-01, zzi= 2.000E-01
     proportional damping= 2.300E+00, critical damping= 1.200E+00
     dx=1, dy=0, dz=0, rx=1, ry=0, rz=0
```

spring create/modify a numbered spring

spring *spring_# options ;*

where

an *option* can be

n1 *node_#*	assign a structure node as the first node
pm1 *pointmass_#*	assign a point mass as the first node
dx1	constrain spring in the x-direction at the first nod
dy1	constrain spring in the y-direction at the first node
dz1	constrain spring in the z-direction at the first node
rx1	constrain spring about the x-axis at the first node
ry1	constrain spring about the y-axis at the first node
rz1	constrain spring about the z-axis at the first node

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

n2 *node_#*	assign a structure node as the second node
pm2	assign a point mass as the second node
dx2	constrain spring in the x-direction at the second node
dy2	constrain spring in the y-direction at the second node
dz2	constrain spring in the z-direction at the second node
rx2	constrain spring about the x-axis at the second node
ry2	constrain spring about the y-axis at the second node
rz2	constrain spring about the z-axis at the second node
sddn *spd_#*	specify the material properties
orop *flag*	assign an orientation option

where *flag* can be:

0	spring/damper acts along the axis
1	deflection/rotations are measured and force/moments applied along the following vector
2	deflection/rotations are measured and force/moments applied along the projection of the spring/damper onto the plane with the following normal
3	deflection/rotations are measured and force/moments applied along the vector between the following two nodes

prflg *flag*	assign a print flag

where *flag* can be:

0	forces are printed in DEFORC file
1	forces are not printed in DEFORC file

ofsi *offset*	assign an initial offset
xco *x-component*	assign a x-component of the orientation vector
yco *y-component*	assign a y-component of the orientation vector
zco *z-component*	assign a z-component of the orientation vector
n3 *node_#*	assign a third node for orientation
n4 *node_#*	assign a fourth node for orientation

Remarks

This command specifies the direction of the spring and the material. The nodes defining the ends of the spring can be from a node in the mesh or a point mass (see **npm** and **pm**). The **spring** command is usually invoked two times to generate a single spring, once for each node of the spring. This can be done across several parts or in the Merge Phase. All of the options are not needed by all output options.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

delspd Delete a numbered spring/damper

delspd *spring/damper-#*

Remarks

The spring/damper may have been defined using the **spring** command or it may have been read into the **True*Grid*®** internal data base using the **readmesh** command.

delspds Delete a list of numbered springs/dampers

delspds *spring/damper_list ;*

10. Element Commands

The recommended method of beam element generation in the merge phase uses the **bm** command. This method of beam generation has the advantage that one can connect any existing noe with any other existing node using a curve or a line. This method has the disadvantage that if a part is changed in a subsequent running of the session file, any reference in the **bm** command to an existing node may be incorrect and will have to be changed. This is the usual, non-parametric, nature of the merge phase. This command is fully interactive. Beams are strung along a 3D curve, interpolated along a line to connect existing nodes.

The second method of beam element generation extracts the needed nodes from an existing shell or brick part. This is only available within the Part Phase and has the usual advantage of the part phase of being parametric. If any parts are changed and the session file rerun, these beam commands will automatically adjust to the changes. The **ibm** and **ibmi** commands create beams along i-lines of the mesh, the **jbm** and **jbmi** along j-lines, and the **kbm** and **kbmi** along k-lines. This is a way to embed beam elements within a shell of a brick structure. Alternatively, the material of the parent shell of brick part can be set to 0 so that the part can be generated as usual, but so that the shell or brick elements will not be saved. The nodes that are used in any of these beam commands will be saved along with the beam elements.

The denigrated **beam** and **cbeam** commands create beams by connecting a pair of nodes with a string of beam elements. Intermediate nodes along the string are linearly interpolated. This is the INGRID method. It is not interactive. It is easy to change an INGRID beam part into this part.

Beam properties are defined using **bsd** and **bind**.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Cross sectional properties, and in particular thicknesses, are not scaled by the **xsca**, **ysca**, **zsca**, and **csca** commands.

bm create a string of beam elements

bm *options ;*

where

option can be:

(Selection of the first node)

n1 *node_#*	to make an existing node the first node of the beams.
pm1 *point_mass_#*	to make a point mass node the first node of the beams.
rt1 *x y z const ;*	to create the first node of the beams in Cartesian coordinates.
cy1 $\rho\ \theta\ z$ *const ;*	to create the first node of the beams in cylindrical coordinates.
sp1 $\rho\ \theta\ \phi$ *const ;*	to create the first node of the beams in spherical coordinates.

(Selection of the second node)

n2 *node_#*	to make an existing node the last node of the beams.
pm2 *point_mass_#*	to make a point mass the last node of the beams.
rt2 *x y z const ;*	to create the last node of the beams in Cartesian coordinates.
cy2 $\rho\ \theta\ z$ *const ;*	to create the last node of the beams in cylindrical coordinates.
sp2 $\rho\ \theta\ \phi$ *const ;*	to create the last node of the beams in spherical coordinates.

(Selection of the orientation)

n3 *node_#*	to make an existing node the last node of the beams.
pm3 *point_mass_#*	to make a point mass the last node of the beams.
rt3 *x y z const ;*	to create the last node of the beams in Cartesian coordinates.
cy3 $\rho\ \theta\ z$ *const ;*	to create the last node of the beams in cylindrical coordinates.
sp3 $\rho\ \theta\ \phi$ *const ;*	to create the last node of the beams in spherical coordinates.
orient *x y z*	to specify a coordinate triple to orient the beams.
sd *surface_#*	to orient beam axes in the orientation of the normal of the surface
v *x y z*	to orient beam axes in the direction of the vector

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

(Misc. options)

mate *material_#* to specify the material number.

cs *cross_section_#* to specify the cross section number (see **bsd**).

nbms *number_of_beams* to specify the number of beams in the string (default is 1).

indc *const ;* to specify the constraints on the intermediate nodes.

cur *3d_curve_#* to interpolate the string of beams along a 3D curve.

(Selection of the nodal spacing)

res *geometricratio* for relative spacing of nodes (default is equal spacing).

drs *first_geometricratio second_geometricratio* for double relative spacing of nodes.

nds *nodal_distribution_function_#* for nodal distribution by a function.

as 0 *first_thickness* first element length

as 1 *last_thickness* last element length

das *first_element_thickness last_element_thickness* first and last element length

sthi *sthi* for thickness in the y-direction.

sthi1 *sthi1* for thickness in the y-direction at the first end point.

sthi2 *sthi2* for thickness in the y-direction at the last end point.

tthi *tthi* for thickness in the z-direction.

tthi1 *tthi1* for thickness in the z-direction at the first end point.

tthi2 *tthi2* for thickness in the z-direction at the last end point.

csarea *csarea* for the cross section area

sharea *sharea* shear area

inertia *Iss Itt Irr* inertia moments

vold *volume* volume of Discrete Beam

lump *inertia* lumped inertia

cablcid *system_#* local coordinate system id number defined by the **lsys**

cabarea *area* cable area

caboff *offset* cable offset

(Selection of the nodal offsets)

noint for no interior node offset interpolation

roff1 *roff1* for x-component of offset vector for first node

soff1 *soff1* for y-component of offset vector for first node

toff1 *toff1* for z-component of offset vector for first node

roff2 *roff2* for x-component of offset vector for last node

soff2 *soff2* for y-component of offset vector for last node

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

toff2 *toff2* for z-component of offset vector for last node

(Selection of the pin flags)

ldr1 release the x-translation constraint at first node
lds1 release the y-translation constraint at first node
ldt1 release the z-translation constraint at first node
lrr1 release the x-rotation constraint at first node
lrs1 release the y-rotation constraint at first node
lrt1 release the z-rotation constraint at first node
ldr2 release the x-translation constraint at last node
lds2 release the y-translation constraint at last node
ldt2 release the z-translation constraint at last node
lrr2 release the x-rotation constraint at last node
lrs2 release the y-rotation constraint at last node
lrt2 release the z-rotation constraint at last node
ldr3 release the x-translation constraint at intermediate nodes
lds3 release the y-translation constraint at intermediate nodes
ldt3 release the z-translation constraint at intermediate nodes
lrr3 release the x-rotation constraint at intermediate nodes
lrs3 release the y-rotation constraint at intermediate nodes
lrt3 release the z-rotation constraint at intermediate nodes
ldp *displacement* for the initial longitudinal displacement.
theta *angle* for the orientation angle for the cross section.
warpage *first_warpage_node second_warpage_node* for two nodes used to determine warpage in the beam.
geom *option* for the method of determining curvature for the NASTRAN CBEND element.

where *option* can be

1 Center of curvature
2 Tangent of Centroid Arc
3 Bend Radius
4 Arc Angle

where *const* can be any of

dx to constrain the x-displacement
dy to constrain the y-displacement
dz to constrain the z-displacement
rx to constrain the x-axis rotation

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

ry	to constrain the y-axis rotation
rz	to constrain the z-axis rotation

Remarks

There are many options to this command. However, many of the options are specific to a single simulation code. There is some overlap, but there is little consistency among the simulation codes on beam element properties. Care must be taken in selecting the options by knowing the options needed for the target simulation code. The dialogue box makes these selections easier.

This command is functional in the Merge Phase, and it is designed to create a general collection of beams or a single beam. We recommend that you use the dialogue box for **bm**.

You can use an existing node of the mesh for a beam, specify coordinates to create a new node for a beam, or you can use a point mass as a node for a beam. Coordinates can be specified in Cartesian, cylindrical, or spherical coordinates.

Beam orientation can be defined using a third node, using a point mass, or by creating another node in Cartesian, cylindrical, or spherical coordinates. Use the output-code specific options in the MATERIAL Menu of the Control Phase to define materials for the beams.

Use the **bsd** to define a beam cross-section type.

Nodes are automatically created if the number of beams specified is greater than 1.

You can define beam elements that follow a 3D curve, and specify the number of such elements, along with a spacing rule for the intermediate nodes.

Optional thickness parameters may be specified for the first and last beams when creating multiple beams. Intermediate beams will have thicknesses that are interpolated from the end beams. You may specify offsets for the first and last nodes, and optionally interpolate these offsets to intermediate nodes.

Constraints which couple the beams to the existing mesh can be eliminated. This may be done separately for the first, last, and intermediate nodes. An initial longitudinal displacement can be specified. An optional orientation angle can be specified. Warpage nodes can be defined for codes which support such options. **Bend** geometry options can be specified for codes which support such options.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

A simple frame structure composed of MARC beam elements is created. At first, the basic framework of beam elements is defined in the Part Phase with use of the **ibmi**, **jbmi** and **kbmi** commands. Then all 4 linear bricks are deleted by the **delem** command (or alternatively you can set the material of bricks to 0). The beams `1 7 8 2 27 26 25` are also deleted to get the desired model. Four new beams `27 28 29 30` are created by the **bm** command using existing nodes. Nodes are labeled in **Figure 231**. Beams are labeled in **Figure 232**. The simplified command file follows:

```
block 1 2 3;1 2 3;1 2;0 5 10;0 5 10;0 5;
     c strucured block mesh definition

ibmi 1 3; ; ;3 2 1 j 1 ;
 c creation of beams in the i-direction 3 columns in the
 c j-direction and 2 columns in the k-direction

jbmi 1 3; ; ;3 2 1 i 1 ;
 c creation of beams in the j-direction 3 columns in the
 c i-direction and 2 columns in the k-direction

kbmi 1 3; ; ;3 3 1 i 1 ;
 c creation of beams in the k-direction 3 columns in the
 c i-direction and 3 columns in the j-direction

merge

delem lb 1 2 3 4;   c linear bricks 1 2 3 4 are deleted

delem lbm 1 7 8 2 27 26 25 ;
     c linear beams 1 7 8 2 27 26 25 are deleted

bsd 1 marc52 area .1 iyy .053 izz .039 ixx .0444
   etsay 100 etsaz 100  ; ;
 c cross section 1 definition
 c for marc52 beam - elastic beam
 c ... cross sectional properties

bm n1 4 n2 12 mate 1 cs 1 ;
 c beam definition by first node (n1) 4 and second node (n2) 12
 c with material 1 and cross section 1

bm n1 12 n2 16 mate 1 cs 1 ;
 c beam definition by first node (n1) 12 and second node (n2) 16
 c with material 1 and cross section 1
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
bm n1 3 n2 11 mate 1 cs 1 ;
 c beam definition by first node (n1) 3 and second node (n2) 11
 c with material 1 and cross section 1

bm n1 11 n2 15 mate 1 cs 1 ;
 c beam definition by first node (n1) 11 and second node (n2) 15
 c with material 1 and cross section 1

labels nodes   c nodes are labeled

labels 1d      c beams are labeled
```

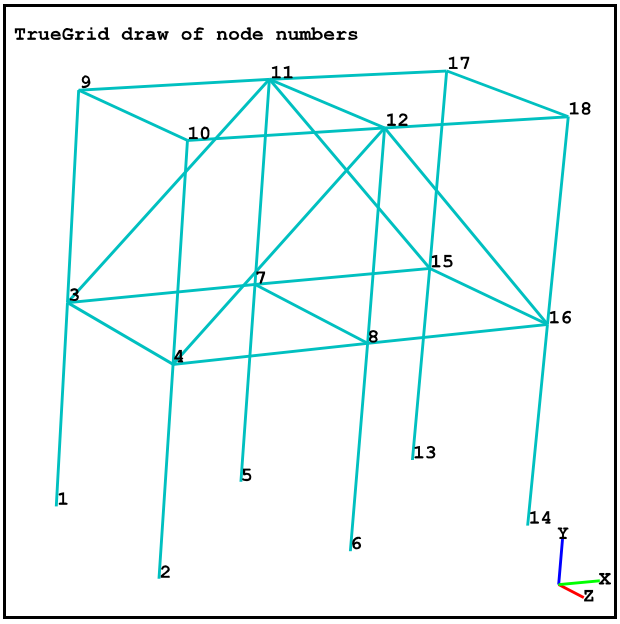


Figure 231 Beam Structure

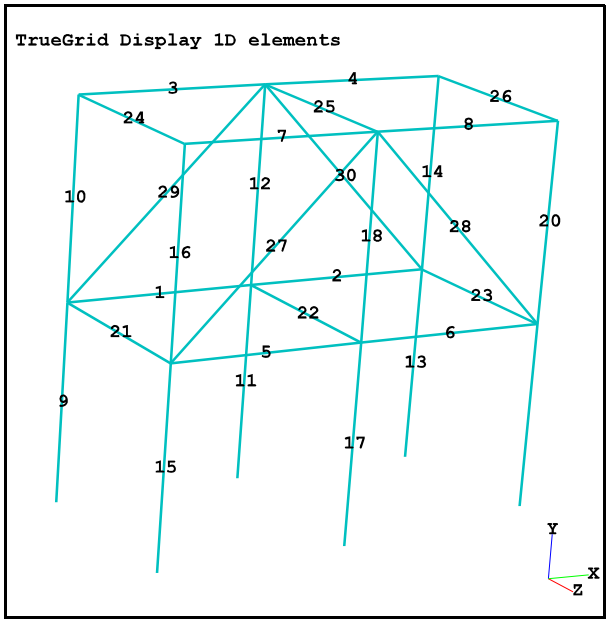


Figure 232 Beam Structure

Example

The circular arc 3d curve is created by the **curd** command. Ten beam elements are generated along the curve by the **bm** command (**Figure 233**). The first and last node coordinates do not have to be coincident with the first and last point of the 3d curve.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
merge

curd 1 arc3
 seqnc rt -10 0 0
       rt 0 4 0
       rt 10 0 0 ;
 c 3d curve definition
 c circular arc by sequence
 c of 3 points

bm rt1 -10 0 0 ; rt2 10 0 0 ;
  mate 1 cs 1 nbms 10 cur 1 ;
 c beam definition
 c -10 0 0 and 10 0 0
 c are coordinates of the
 c first and last point
 c on arc
 c material 1 and cross
 c section 1 are assigned
 c to all 10 beams
 c 3d curve  number 1 is used

labels 1d      beams are labeled
```

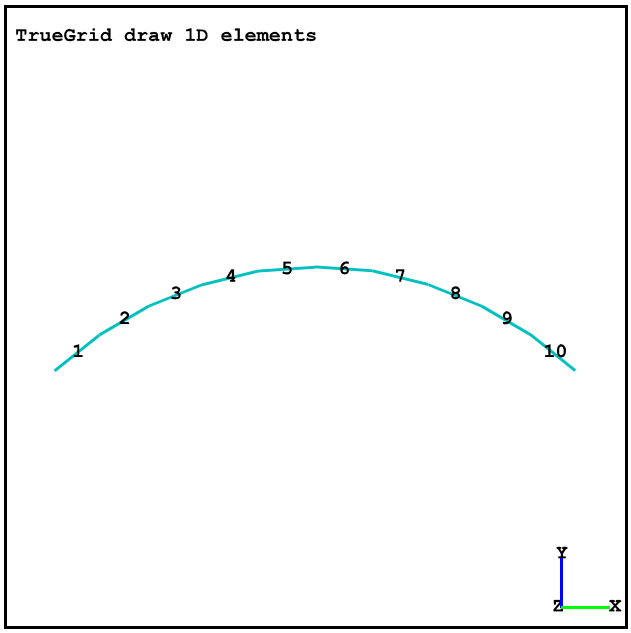


Figure 233 beams by bm

Example

The circular arc 3d curve is created by the **curd** command. Ten beam elements are generated along the curve by the **bm** command (**Figure 234**) by geometrical spacing with a factor of 0.9

```
merge
curd 1 arc3
 seqnc rt -10 0 0
       rt 0 4 0
       rt 10 0 0 ;
 c 3d curve definition
 c circular arc by sequence
 c of 3 points
bm rt1 -10 0 0 ; rt2 10 0 0 ;
  mate 1 cs 1 nbms 10 cur 1
  res .9;
 c Beam definition
 c -10 0 0 and 10 0 0
 c are coordinates of the
```

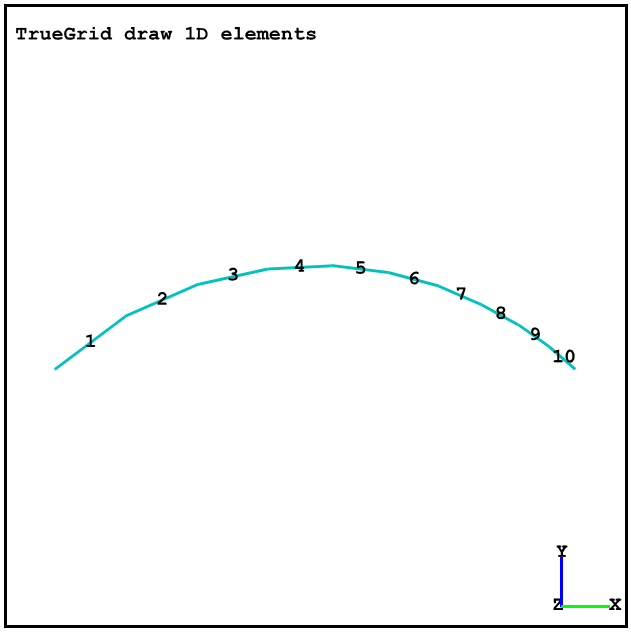


Figure 234 beams by bm

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
 c first and last node.
 c Material 1 and cross
 c section 1 are assigned
 c to all 10 beams
 c 3d curve  number 1 uses
 c the spacing factor .9

labels 1d      beams are labeled
```

Example

Twenty beams are created by 2 **bm** commands from coordinates of the start and end nodes. The start and end nodes have assigned boundary conditions in x, y and z- directions. The local axes of all beams are pointing towards the same orientation node with coordinates `(.5,.5,0.)`. The intermediate nodes (marked by red circles in **Figure 235**) have constrained displacements in the z-direction. The simplified command file follows:

```
merge
bm rt1 0 0 0 dx dy dz ;
   rt2 1 0 0  ; rt3 .5 .5 0. ;
   mate 1 cs 2 nbms 10
   indc dz ;;
 c beam definition
 c 0 0 0 and 1 0 0
 c are coordinates of the
 c first and last node
 c .5 .5 0 are coordinates
 c of the orientation node
 c material 1 and cross
 c section 2 are assigned
 c to all 10 beams
 c the intermediate nodes have
 c constrained displacements
 c dz
bm rt1 1 0 0  ;
   rt2 1 1 0 dx dy dz ;
   rt3 .5 .5 0. ;
  mate 1 cs 2 nbms 10
  indc dz ; ;
 c beam definition
 c 1 0 0 and 1 1 0
 c are coordinates of the
 c first and last node
 c .5 .5 0 are coordinates
 c of the orientation node
 c material 1 and cross
```

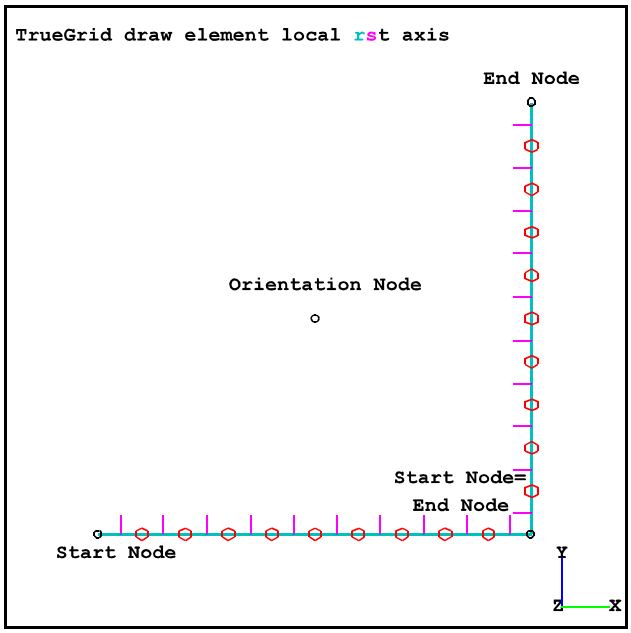


Figure 235 orientation of beams

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
 c section 2 are assigned
 c to all 10 beams
 c the intermediate nodes have
 c constrained displacements
 c dz
```

Example

A beam structure is created by the **bm** command. A parametric 3d curve representing a parabola is created by the **curd** command. Twenty beams are generated along the 3d curve. The vertical beams (numbers `21` through `53`) are generated from the start nodes numbered `4, 6, 8, 10, 12, 14, 16, 18,` and `20` respectively. The coordinates of the last node in each vertical string of beams are formed from the coordinates of the first node, respectively, by zeroing the y-coordinate. It is useful to use ***Pick*** by ***Label*** in the Environment Window and F7 to fill the coordinates of a node into a dialogue box. Beams of the base (numbers `54` through `73`) are generated from the pairs of existing nodes consecutively (**Figure 236** and **Figure 237**). The command file follows:

merge

```
curd 1 3dfunc -10 10 u ; -u*u/25+4 ;0; ;
 c 2D curve parametric definition - parabola

bm rt1 -10 0 0 ; rt2 10 0 0 ; mate 3 cs 2 nbms 20 cur 1 ;
 c 20 beams are generated on the curve 1 - parabola

bm n1 4  rt2 -8.2257433e+00 0.0 0.0 ; mate 2 cs 1 nbms 2 ;
bm n1 6  rt2 -6.3288207e+00 0.0 0.0 ; mate 2 cs 1 nbms 3 ;
bm n1 8  rt2 -4.3093190e+00 0.0 0.0 ; mate 2 cs 1 nbms 4 ;
bm n1 10 rt2 -2.1856163e+00 0.0 0.0 ; mate 2 cs 1 nbms 5 ;
bm n1 12 rt2 -2.3855813e-07 0.0 0.0 ; mate 2 cs 1 nbms 5 ;
bm n1 14 rt2  2.1856163e+00 0.0 0.0 ; mate 2 cs 1 nbms 5 ;
bm n1 16 rt2  4.3093190e+00 0.0 0.0 ; mate 2 cs 1 nbms 4 ;
bm n1 18 rt2  6.3288207e+00 0.0 0.0 ; mate 2 cs 1 nbms 3 ;
bm n1 20 rt2  8.2257433e+00 0.0 0.0 ; mate 2 cs 1 nbms 2 ;
 c The vertical beams (numbers 21 through 53) are generated
 c from the start node numbers
 c 4, 6, 8, 10, 12, 14, 16, 18, 20 and
 c last node coordinates,
 c which are derived from the start ones by zeroing
 c the y-coordinate. Number of beams is from 2 to 5

bm n1  1 n2 22 mate 2 cs 3 nbms 2 ;
bm n1 22 n2 24 mate 2 cs 3 nbms 2 ;
bm n1 24 n2 27 mate 2 cs 3 nbms 2 ;
bm n1 27 n2 31 mate 2 cs 3 nbms 2 ;
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
bm n1 31 n2 36 mate 2 cs 3 nbms 2 ;
bm n1 36 n2 41 mate 2 cs 3 nbms 2 ;
bm n1 41 n2 46 mate 2 cs 3 nbms 2 ;
bm n1 46 n2 50 mate 2 cs 3 nbms 2 ;
bm n1 50 n2 53 mate 2 cs 3 nbms 2 ;
bm n1 53 n2  2 mate 2 cs 3 nbms 2 ;
 c Beams of the base (numbers 54 through 73) are generated
 c from the pairs of existing nodes, consecutively

labels 1d      c beams are labeled

labels nodes        c nodes are labeled
```

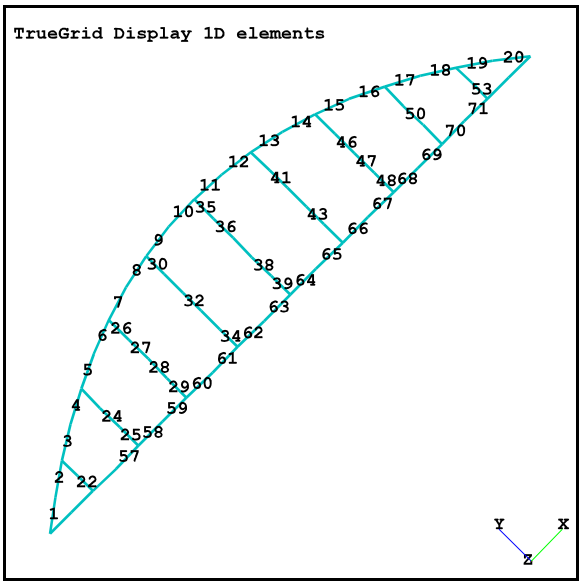


Figure 236 beams by bm

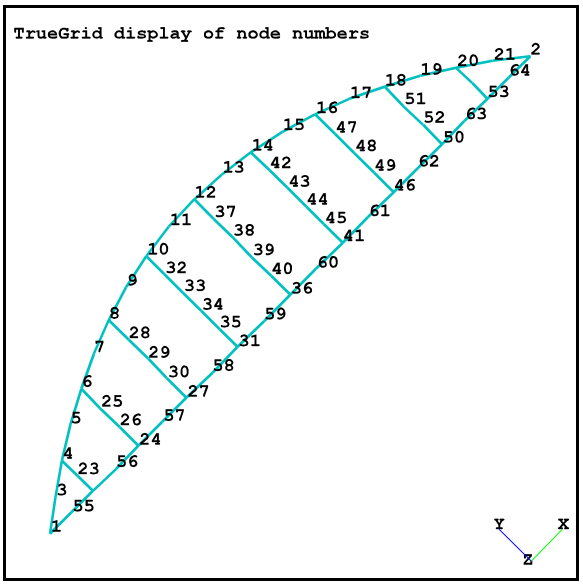


Figure 237 beams by bm

bms change the section properties of a set of beams

bms *selection options ;*

where

selection can be

set element_set	select an element set containing beam elements
lbm element_list ;	select a list of beam elements by number

option can be:

orient *x y z*	specify a coordinate triple to orient the beams.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

v *x y z*	orient beam axes in the direction of the vector
cs *cross_section_#*	specify the cross section number (see **bsd**).
sthi *sthi*	thickness in the y-direction.
sthi1 *sthi1*	thickness in the y-direction at the first node
sthi2 *sthi2*	thickness in the y-direction at the last node
tthi *tthi*	thickness in the z-direction.
tthi1 *tthi1*	thickness in the z-direction at the first node
tthi2 *tthi2*	thickness in the z-direction at the last node
csarea *csarea*	the cross section area
sharea *sharea*	shear area
inertia *Iss Itt Irr*	inertia moments
vold *volume*	volume of Discrete Beam
lump *inertia*	lumped inertia
cablcid *system_#*	local coordinate system id number defined by **lsys**
cabarea *area*	cable area
caboff *offset*	cable offset
noint	for no interior node offset interpolation
roff1 *roff1*	for x-component of offset vector for first node
soff1 *soff1*	for y-component of offset vector for first node
toff1 *toff1*	for z-component of offset vector for first node
roff2 *roff2*	for x-component of offset vector for last node
soff2 *soff2*	for y-component of offset vector for last node
toff2 *toff2*	for z-component of offset vector for last node
ldr1	release the x-translation constraint at 1st node
lds1	release the y-translation constraint at 1st node
ldt1	release the z-translation constraint at 1st node
lrr1	release the x-rotation constraint at 1st node
lrs1	release the y-rotation constraint at 1st node
lrt1	release the z-rotation constraint at 1st node
ldr2	release the x-translation constraint at 2nd node
lds2	release the y-translation constraint at 2nd node
ldt2	release the z-translation constraint at 2nd node
lrr2	release the x-rotation constraint at 2nd node
lrs2	release the y-rotation constraint at 2nd node
lrt2	release the z-rotation constraint at 2nd node
ldr1o	no release of the x-translation constraint at 1st node
lds1o	no release of the y-translation constraint at 1st node
ldt1o	no release of the z-translation constraint at 1st node
lrr1o	no release of the x-rotation constraint at 1st node
lrs1o	no release of the y-rotation constraint at 1st node

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

lrt1o no release of the z-rotation constraint at 1st node
ldr2o no release of the x-translation constraint at 2nd node
lds2o no release of the y-translation constraint at 2nd node
ldt2o no release of the z-translation constraint at 2nd node
lrr2o no release of the x-rotation constraint at 2nd node
lrs2o no release of the y-rotation constraint at 2nd node
lrt2o no release of the z-rotation constraint at 2ns node
ldp *displacement* for the initial longitudinal displacement.
theta *angle* for the orientation angle for the cross section.
warpage *first_warpage_node second_warpage_node*
two nodes used to determine warpage in the beam.
geom *option* for the method of determining curvature for the NASTRAN CBEND element.

where *option* can be

1 Center of curvature
2 Tangent of Centroid Arc
3 Bend Radius
4 Arc Angle

where *const* can be any of

dx to constrain the x-displacement
dy to constrain the y-displacement
dz to constrain the z-displacement
rx to constrain the x-axis rotation
ry to constrain the y-axis rotation
rz to constrain the z-axis rotation

Remarks

Refer to the bm command for more details on the meaning of these cross sectional properties.

delem delete a set of elements

delem *option list_of_elements ;*

where an

option must be one of the following:

lb for 3D linear 8-node brick elements
ls for 2D linear 4-node shell elements
lbm for 1D linear 2-node beam elements
qb for 3D quadratic 8-node brick elements

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

qs for 2D quadratic 4-node shell elements
eset *set_name* to delete all elements in an element set

Remarks

All boundary constraints and conditions associated with these deleted elements are removed from the **True*Grid*®** internal database. Any nodes used to define these elements, which are not used in the definition of other elements, are removed. The sorting and data rearrangement due to element deletion can be extensive and time consuming. For this reason, it is recommended that a group of elements be deleted instead of one at a time. This command cannot be undone. When elements are deleted, they cannot be retrieved.

etd specify the element types to be displayed in the graphics

etd options ;
where
an *option* can be any of the following:
1dl *switch* for quadratic 1D elements
2dl *switch* for quadratic 2D elements
3dl *switch* for quadratic 3D elements
2dq *switch* for quadratic 2D elements
3dq *switch* for quadratic 3D elements
where
switch can be
on
off

rbe rigid body elements

rbe *element_# type parameters*
where the *parameters* vary depending on the *type* from the following
rrod *node const ; node const ;*
where only one of the following constraints can be selected for *const*
mdx dependent translation in x
mdy dependent translation in y
mdz dependent translation in z
where a *node* can be selected by it's number or by coordinates
node node_# any exist node
rt *x y z* new node in Cartesian coordinates

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

cy *rho theta z* new node in cylindrical coordinates
sp *rho theta phi* new node in spherical coordinates

rbar *node const ; node const ;*

where a constraint *const* can be formed using

ndx independent translation in x
ndy independent translation in y
ndz independent translation in z
nrx independent rotation in x
nry independent rotation in y
nrz independent rotation in z
mdx dependent translation in x
mdy dependent translation in y
mdz dependent translation in z
mrx dependent rotation in x
mry dependent rotation in y
mrz dependent rotation in z

where a *node* can be selected by it's number or by coordinates

node node_# any exist node
rt *x y z* new node in Cartesian coordinates
cy *rho theta z* new node in cylindrical coordinates
sp *rho theta phi* new node in spherical coordinates

rtrplt *node const ; node const ; node3 const ;*

where a constraint *const* can be formed using

ndx independent translation in x
ndy independent translation in y
ndz independent translation in z
nrx independent rotation in x
nry independent rotation in y
nrz independent rotation in z
mdx dependent translation in x
mdy dependent translation in y
mdz dependent translation in z
mrx dependent rotation in x
mry dependent rotation in y
mrz dependent rotation in z

where a *node* can be selected by it's number or by coordinates

node node_# any exist node
rt *x y z* new node in Cartesian coordinates
cy *rho theta z* new node in cylindrical coordinates
sp *rho theta phi* new node in spherical coordinates

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

rbe2 *node const ; list_nodes ;*

where a constraint *const* can be formed using

mdx	dependent translation in x
mdy	dependent translation in y
mdz	dependent translation in z
mrx	dependent rotation in x
mry	dependent rotation in y
mrz	dependent rotation in z

where a *node* can be selected by it's number or by coordinates

node node_#	any exist node
rt *x y z*	new node in Cartesian coordinates
cy *rho theta z*	new node in cylindrical coordinates
sp *rho theta phi*	new node in spherical coordinates

rbe3 *node const ; [node const ; weight list_nodes ;];*

where a constraint *const* can be formed using

mdx	dependent translation in x
mdy	dependent translation in y
mdz	dependent translation in z
mrx	dependent rotation in x
mry	dependent rotation in y
mrz	dependent rotation in z

where a *node* can be selected by it's number or by coordinates

node node_#	any exist node
rt *x y z*	new node in Cartesian coordinates
cy *rho theta z*	new node in cylindrical coordinates
sp *rho theta phi*	new node in spherical coordinates

Remarks

This command generates rigid body elements for NASTRAN and NE/NASTRAN. The nodes used to form the rigid body are identified by node number or by coordinates in the neighborhood. This command supports 5 types of rigid bodies. They are:

1. **rrod** is for a rob rigid element formed with two nodes with one end pinned.
2. **rbar** is for a bar rigid element formed with two nodes and 6 DOF.
3. **rtrplt** is for a plate rigid element formed with three nodes.
4. **rbe2** is for a rigid body.
5. **rbe3** is for an interpolation constraint element.

The dependencies are best described in the NASTRAN or NE/NASTRAN manuals.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

11. Part Types

These part type commands have no affect on the models imported using the **readmesh** command.

linear specify following parts to use linear elements

linear (no arguments)

Remarks

For all subsequent parts, all brick elements with have 8 nodes, one at every corner of the brick. Shell elements will have 4 nodes, one at every corner. Beam elements will have 2 nodes, one at each end. This is the default.

After entering the merge phase and merging nodes, if an edge or face becomes degenerate, forming a different element type, such as a wedge or tetrahedron, the number of nodes are changed for that element.

quadratic specify following parts to use quadratic elements

quadratic (no arguments)

Remarks

For all subsequent parts, all brick elements with have 20 nodes, one at every corner of the brick and an additional mid-edge node for each of the 12 edges of the element. Shell elements will have 4 nodes, one at every corner and an additional mid-edge node for each of the 4 edges of the shell element. Beam elements will have 2 nodes, one at each end.

After entering the merge phase and merging nodes, if an edge or face becomes degenerate, forming a different element type, such as a wedge or tetrahedron, the number of nodes are changed for that element.

partmode part command indices format

partmode *mode*

where

mode can be

s	for standard (default)
i	for interval

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

The standard method is described in the **block**, **cylinder**, and **blude** command remarks. The interval mode uses 3 lists of numbers, one for each index direction. The difference is that they are the number of elements for each region. This implies that there will be one less number in each list if the interval mode is specified. An analogy is a fence. The standard method indicates the position of each fence post from the start of the fence while the interval mode specifies gaps between the fence posts.

This command only applies to the **block**, **cylinder**, and **blude** commands.

The default is s.

The standard mode is more complex but has the advantage of creating both solids and shells. It also has the same format as the index progressions in the part phase commands. The interval mode is preferred because if the number of elements in one region is to be modified, only one number in the part command needs to be modified.

Examples

In these two examples, the result is the same.

```
parameters
        i1 1
        i2 [%i1+2]
        i3 [%i2+4]
        j1 1
        j2 [%j1+4]
        k1 1
        k2 [%k1+1]
        k3 [%k2+2] ;
partmode s c default
block %i1 %i2 %i3;%j1 %j2;%k1 %k2 %k3;0 .1 .2;1 1.2;2 2.3 2.5;

partmode i
block 2 4;4;1 2;0 .1 .2;1 1.2;2 2.3 2.5;
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

12. Displacements, Velocities, and Accelerations

These commands specify displacements, velocities, and accelerations, usually for initial or boundary conditions. In most of them, the arguments take the form

load_curve amplitude x y z

The displacement, velocity, etc. is applied in the direction given by the vector (*x*, *y*, *z*), which might be in Cartesian, cylindrical, or spherical coordinates. For some simulation codes like DYNA3D, the magnitude of the condition is the *product* of the *amplitude* and the *current value of the load curve*. In this case, the load curve is a time-dependent function given by the load curve number, *load_curve*. In some other simulation codes, such as ABAQUS, the load curve number is associated with a step, and in other simulation codes like NASTRAN, the load curve number is associated with a load case.

Also see the Displacement, Velocities, and Acceleration section in the Part Phase (Generation chapter).

acc Cartesian prescribed nodal boundary acceleration

acc *nodes load_curve ampl options x y z*

where

nodes can be

n *node_number*	for a node number
rt *x y z*	for a point in Cartesian coordinates
cy *rho theta z*	for a point in cylindrical coordinates
sp *rho theta phi*	for a point in spherical coordinates
nset *name_of_set*	for a node set

load_curve	for a load curve or zero
ampl	for the amplitude

where *options* can be any of

exclude	exclude normal directions (for Lsdyna)
birth *time*	specify starting time (for Lsdyna)
death *time*	specify ending time (for Lsdyna)

where

x y z	for the x, y, and z components of the acceleration

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

bv prescribed boundary surface velocities for NEKTON

bv fset *face_set fx fy fz*

deform assigns deformations to beam elements

deform *type object sid deformation*

where *type* and *object* are related

beam *beam_#*	beam number (rods or bars)
eset *element_set*	element set name, assumed to contain rods or bars
where *sid*	set identification number
where *deformation*	magnitude of deformation

Remarks

This command defines rod and bar element deformations for NASTRAN and NE/NASTRAN. It does not check to see if the beam element cross section type (**bsd**) is a rod or bar.

dis initial nodal displacement

dis *nodes option x y z*

where

nodes can be

n *node_number*	for a node number
rt *x y z*	for a point in Cartesian coordinates
cy *rho theta z*	for a point in cylindrical coordinates
sp *rho theta phi*	for a point in spherical coordinates
nset *name_of_set*	for a node set

option can be

sid *id*	for a set identification number
r	to flag this command for rotational conditions
x y z	for the x, y, and z-components of the displacement

Remarks

When a node is assigned several displacements, the last one specified is the one that is used. The **r** flag converts this command to mean nodal rotational velocities. The **sid** is used by several output options to indicate a load case or step.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

fd displacement boundary condition

fd *nodes load_curve ampl options x y z*

where

nodes can be

n *node_number*	for a node number
rt *x y z*	for a point in Cartesian coordinates
cy *rho theta z*	for a point in cylindrical coordinates
sp *rho theta phi*	for a point in spherical coordinates
nset *name_of_set*	for a node set

load_curve	for a load curve or zero
ampl	for the amplitude

where *options* can be any of

exclude	exclude normal directions (for Lsdyna)
birth *time*	specify starting time (for Lsdyna)
death *time*	specify ending time (for Lsdyna)

where

x y z	are the x, y, and z-components of the displacement.

fv prescribed velocities

fv *nodes load_curve ampl options x y z*

where

nodes can be

n *node_number*	for a node number
rt *x y z*	for a point in Cartesian coordinates
cy *rho theta z*	for a point in cylindrical coordinates
sp *rho theta phi*	for a point in spherical coordinates
nset *name_of_set*	for a node set

load_curve	for a load curve number or zero
ampl	for the amplitude

where *options* can be any of

exclude	exclude normal directions (for Lsdyna)
birth *time*	specify starting time (for Lsdyna)
death *time*	specify ending time (for Lsdyna)

where

x y z	for the x, y, and z-components of the velocity

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

frb prescribed rotational boundary

frb *nodes load_# amplitude options condition direction*

where *nodes* can be

n *node_number*
rt *x y z*
cy $\rho\ \theta\ z$
sp $\rho\ \theta\ \phi$
nset *set_name*

where *options* can be any of the following

birth *time*
death *time*
o**ffset** $offset_1\ offset_2$

where *condition* must be one of the following

v	velocities
a	accelerations
d	displacements
dofv	nodal dof velocities
dofa	nodal dof accelerations
dofd	nodal dof displacements

where *direction* must be one of the following

x	about the x-axis
y	about the y-axis
z	about the z-axis
v $x_0\ y_0\ z_0$	about an arbitrary axis
ex	not about the x-axis
ey	not about the y-axis
ez	not about the z-axis
ev $x_0\ y_0\ z_0$	not about an arbitrary axis

Remarks

A condition can be a velocity, acceleration, displacement, or a nodal rotation. This is suited for Dyna3D (velocities, accelerations, and nodal rotations) and Lsdyna. In these codes, the selected nodes are prescribed this condition relative to an axis of rotation.

Use the **frb** option in the **co** merge phase command in the merge phase to display these conditions.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

fvv prescribed variable velocities

fvv *nodes load_curve amp_expr ; x_expr ; y_expr ; z_expr ;*

where

nodes can be

n *node_number*	for a node number
rt *x y z*	for a point in Cartesian coordinates
cy *rho theta z*	for a point in cylindrical coordinates
sp *rho theta phi*	for a point in spherical coordinates
nset *name_of_set*	for a node set

load_curve	for a load curve number or zero
amp_expr	for the FORTRAN amplitude
x_expr	for the FORTRAN expression for the velocity x-component
y_expr	for the FORTRAN expression for the velocity y-component
z_expr	for the FORTRAN expression for the velocity z-component

Remarks

This command assigns velocities, allowing for the amplitude factor and the Cartesian vector components to be calculated using a FORTRAN expression. Each expression can reference the nodal coordinates x, y, and z.

vacc Cartesian prescribed variable nodal boundary acceleration

vacc *nodes load_curve amp_expr ; x_expr ; y_expr ; z_expr ;*

where

nodes can be

n *node_number*	for a node number
rt *x y z*	for a point in Cartesian coordinates
cy *rho theta z*	for a point in cylindrical coordinates
sp *rho theta phi*	for a point in spherical coordinates
nset *name_of_set*	for a node set

load_curve	for a load curve number or zero
amp_expr	for the FORTRAN amplitude
x_expr	for the FORTRAN expression for the x-component of the acceleration
y_expr	for the FORTRAN expression for the y-component of the acceleration
z_expr	for the FORTRAN expression for the z-component of the acceleration

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

This command assigns accelerations, allowing for the amplitude factor and the Cartesian vector components to be calculated using a FORTRAN expression. Each expression can reference the nodal coordinates X, Y, and Z.

ve initial nodal velocity

ve *nodes option x y z*

where

nodes can be

n *node_number*	for a node number
rt *x y z*	for a point in Cartesian coordinates
cy *rho theta z*	for a point in cylindrical coordinates
sp *rho theta phi*	for a point in spherical coordinates
nset *name_of_set*	for a node set

option can be

sid *id*	for a set identification number
r	to flag this command for rotational conditions
x y z	for the x, y, and z-components of the velocity

Remarks

This overrides the velocities specified by the **velocity** and **rotation** commands. When a node is assigned several velocities, the last one specified is the one that is used. The **r** flag converts this command to mean nodal rotational velocities. The **sid** is used by several output options to indicate a load case or step.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

13. Force, Pressure, and Loads

These commands specify loads, displacements, or similar conditions. In most of them, the arguments include

load_curve amplitude x y z

The load, displacement, etc. is applied in the direction given by the vector (*x*, *y*, *z*), which might be in Cartesian, cylindrical, or spherical coordinates. For some simulation codes like DYNA3D, the magnitude of the load, displacement, etc. is the *product* of the *amplitude* and the *current value of the load curve*. In this case, the load curve is a time-dependent function given by the load curve number, *load_curve*. In some other simulation codes, such as ABAQUS, the load curve number is associated with a step, and in other simulation codes like NASTRAN, the load curve number is associated with a load case.

fa fixed nodal rotations

fa *nodes angle_x angle_y angle_z*

where

nodes can be

n *node_number*
rt *x y z*
cy *rho theta z*
sp *rho theta phi*
nset *name_of_set*

angle_x is the rotation about the x axis,
angle_y is the rotation about the y axis,
angle_z is the rotation about the z axis

fc concentrated nodal loads

fc *nodes load_# amplitude fx fy fz*

where

nodes can be

n *node_number*
rt *x y z*
cy *rho theta z*
sp *rho theta phi*

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

nset *name_of_set*

load_#	is a load curve number, specifying the force's change in time,
amplitude	is a load curve multiplier,
fx	is the initial force in the x-direction
fy	is the initial force in the y-direction
fz	is the initial force in the z-direction

ffc concentrated nodal load with a follower force

ffc *nodes load_# amplitude* $node_1$ $node_2$ $node_3$

where *nodes* can be

n *node_number*
rt *x y z*
cy $\rho\ \theta z$
sp $\rho\ \theta\ \phi$
nset *set_name*

Remarks

A follower plane is defined using the three nodes. The normal of the plane defines the direction of the force. This is used for Dyna3D and Lsdyna.

Use the **co** merge phase command with the **fcc** option to view this condition.

fmom follower nodal moment

fmom *nodes load_# amplitude* $node_1$ $node_2$ $node_3$

where *nodes* can be

n *node_number*
rt *x y z*
cy $\rho\ \theta z$
sp $\rho\ \theta\ \phi$
nset *set_name*

Remarks

A follower plane is defined using three nodes. The moment is about the normal of the follower plane.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

This is used for Lsdyna.

Use the **co** merge phase command with the **fmom** option to view this condition.

mom nodal moment about one global coordinate axis

mom *nodes load_# moment direction*

where

nodes can be

n node_number
rt *x y z*
cy *rho theta z*
sp *rho theta phi*
nset *name_of_set*

direction can be

x,
y, or
z axis

pr pressure load

pr fset *face_set load_curve amplitude*

where

face_set	face set name
load_curve	is a load curve number or zero
amplitude	is an amplitude factor

Remarks

Pressure is a scalar quantity applied to a face of an element. A positive pressure acts on a face in the direction opposite the positive normal of the face.

All faces within the set are assigned the same pressure condition. When a load curve accompanies the condition, the pressure becomes time dependent. If the load curve number is zero, no load curve is specified and the pressure load is considered a constant.

The positive normal direction can be specified using the **orpt** command.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

14. Boundary and Constraint conditions

b nodal displacement and rotation constraints

b *nodes option_list ;*

where

nodes can be

n *node_number*	for a node
rt *x y z*	for a point in Cartesian coordinates
cy *rho theta z*	for a pont in cylindrical coordinates
sp *rho theta phi*	for a point in spherical coordinates
nset *name_of_set*	for a node set

an *option* is

sid *n*	set identification number
dx	for x-displacement
dy	for y-displacement
dz	for z-displacement
rx	for rotation about the x-axis
ry	for rotation about the y-axis
rz	for rotation about the z-axis

followed by a value of

0	in order to initialize to no constraint
1	to constrain

Remarks

This command adds nodal constraints in the global coordinate system. Initially there are no constraints. Each **b** command modifies the constraints for the nodes of the region. Thus several commands may set different constraints for the same node. This has a cumulative effect. For example, you can remove degrees of freedom in the **x**-direction for nodes of an edge of the mesh. Then the constraint lists of all nodes along this edge are modified to reflect this constraint. Then you could place a displacement constraint in the **y**-direction on an adjoining edge of the mesh. The corner node where these two edges meet would then be supported in both the **x** and **y**-directions.

Several other commands can affect the constraints. For example, the **plan** command with the **symm** option for symmetry can add constraints if the symmetry plane is parallel to one of the coordinate planes.

Different nodes may be merged into one. The merged node inherits ALL of the constraints of the nodes which were merged into it. To view the different constraints in the model, use the **condition**

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

command with the **dx**, **dy**, **dz**, **rx**, **ry**, or **rz** options.

Sid specifies a set identification number so that the nodal constraints are written to the NASTRAN and NE/NASTRAN output using the SPC1 and SPCADD keywords. For ABAQUS output, the set identification number becomes the load set number used in abcload option of the **abaqstep** to associate the boundary condition with a step in the analysis.

cfc boundary conditions for the CF3D output option

cfc *id type parameters faces*

where

the *type* and *parameters* can be:

fv *vx vy vz*
ft *temperature*
fsp *species_# amplitude*
ol *pressure*
il *vx vy vz*
wall
ufl *amplitude*
vfl *amplitude*
wfl *amplitude*
tfl *temperature*
spf *species_# amplitude*
cb

faces can be a sequence of the following terminated with a ";"

lb1 *list* ; for linear brick face 1
lb2 *list* ; for linear brick face 2
lb3 *list* ; for linear brick face 3
lb4 *list ;* for linear brick face 4
lb5 *list ;* for linear brick face 5
lb6 *list ;* for linear brick face 6

list is a sequence of element numbers

or

faces can be one of the following

surface *surface_# tolerance #_nodes*

where

tolerance is the accepted distance from the surface
#_nodes is the required number of corner nodes within tolerance

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

set *face_set_name*

Remarks

A set of faces can be generated using options in the Part Phase with the **fset** command. Face sets can be modified or generated in the Merge Phase with either the **fset** command or the set window in the pick panel of the environment window. Only the linear brick element faces will be used by this command. It is usually better to define face sets in the Part Phase so that it is parametric. However, the **cfc** command is also available in the Part Phase so if you are going to define the set in the Part Phase to be used for the **cfc** command, it is simpler to use the **cfc** command in the Part Phase. In those cases when the regions in the part do not create the desired effect, use the set feature in the Merge Phase.

The surface option applies only to the nodes in the picture. This is all of the nodes in the active list of parts and materials.

Examples

```
cfc first fv 1.2 2.3 3.4 surface 33 .001 4
```

This example creates a fixed velocity boundary condition called first. All faces of the mesh with all 4 nodes within .001 distance of surface 33 will be assigned this condition.

```
cfc house ob set hs1
```

This example assigns the obstruction boundary condition called house to all of the faces in the hs1 face set.

```
cfc 3142 ol -2.34e+08 lb1 4 219 410:435; lb3 3:8 29:34;;
```

This command assigns the outlet pressure condition numbered 3142 to face number 1 of the linear brick elements 4, 219, and 410 thru 435. It assigns the same condition to face number 3 of the linear brick elements 3 thru 8 and 29 thru 34.

fbc FLUENT boundary conditions

fbc fset *face_set type zone*
where *type* can be
interior
wall

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

pr_inlet	pressure-inlet
inlet_ve	inlet-vent
intake_f	intake-fan
pr_outle	pressure-outlet
exhaust_	exhaust-fan
outlet_v	outlet-vent
symmetry	
per_shad	periodic-shadow
pr_far_f	pressure-far-field
velocity	velocity-inlet
periodic	
fan	
porous_j	porous-jump
radiator	
mass_flo	mass-flow-inlet
interfac	interface
outflow	
axis	

il identifies an inlet for fluid flow.

il fset *face_set*

infol print nodal information with a specific load/condition

infol *nodes type*

where

nodes can be

n *node_number*	a node number
rt *x y z*	a Cartesian coordinate
cy *rho theta z*	a point in cylindrical coordinates
sp *rho theta phi*	a point in spherical coordinates
nset *name_of_set*	a node set

type can be

all *load_curve_#*	for all nodal conditions
fc *load_#*	for force
fd *load_curve_#*	for fixed displacement
fv *load_curve_#*	for fixed velocity

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

ft *load_curve_#*	for forced temperature
tm	for initial temperature
sw *stone_wall_#*	for stone wall
v	for electrostatic potential
acc *load_curve_#*	for acceleration
fa	for fixed angular displacement
mp	for magnetic potential
mom *load_#*	for moment
lb *local_system_#*	for constraint in local coordinate system
ve	for velocity
te	for temperature
si *interface_#* ***S***	for nodal sliding interface

tepro *load_curve_# scale base_temp* for Temperature

where load_curve_#, stone_wall_#, interface_#, or local_system_# can be 0 for all cases.

Remarks

This command will print a table of nodes with specific loads and conditions to the Text Window. The set of nodes can be selected by node number, location, or by containment in a set. The **b** command is not included since the **conditions** command can graphically display the nodes of a specific constraint.

Some of the types of loads include a load case or load curve number. If 0 is used, then all cases will be used in the command.

jt assign nodes to a joint

jt *joint_# local_node_# options* ;

where *local_node* depends on the type of joint (see **jd**)

where an *option* can be

n	node_number	use an existing node
p *x y z rigid_body_material_#*		new node in Cartesian coordinates
c *rho theta z rigid_body_material_#*		new node in cylindrical coordinates
s *rho theta phi rigid_body_material_#*		new node in spherical coordinates
v *x_offset y_offset z_offset*		offset the new node
dx		x-displacement constraint on new node
dy		y-displacement constraint on new node
dz		z-displacement constraint on new node
rx		x-rotation constraint on new node

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

ry	y-rotation constraint on new node
rz	z-rotation constraint on new node

Remarks

This command assigns a node to a numbered joint defined by **jd**. Each node in a joint is assigned a node sequence number which is referred to as its local node number. Each type of joint requires a different number of nodes and the role a node plays in the joint depends on the joint type and the node local node number. The simplest example of a joint is when a set of nodes share constraints. In this case the ordering of the nodes are not important. Up to 16 nodes can be included in a shared constraint joint. The nodal constraints available in this command apply to new nodes defined using the p, c, or s options.

lb local nodal displacement and rotation constraints

lb *nodes system_# option_list ;*

where a *system* is defined using the command **lsys**

where an *option* is

dx *flag*	for x-displacement
dy *flag*	for y-displacement
dz *flag*	for z-displacement
rx *flag*	for rotation about the x-axis
ry *flag*	for rotation about the y-axis
rz *flag*	for rotation about the z-axis

where the *flag* can be

0	for initialize to no constraint
1	for constrain

where *nodes* must be one of

n *node*	for an existing node
rt *x y z*	for a new node in Cartesian coordinates
cy *rho theta z*	for a new node in cylindrical coordinates
sp *rho theta phi*	for a new node in spherical coordinates
nset *name_of_set*	for a node set

Remarks

Use this command to set constraints on nodes that cannot be set using the **b** or **bi** commands because they are restricted to the global coordinate system. Care is needed not to over specify the constraints on a node. No warnings are given if a node is over constrained. Use the **lsys** command to define the local coordinate system.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

mpc shared nodal (multiple point) constraints for a nodal set

mpc *node_set_name constraints ;*

where

a *constraint* can be:

dx	for constrained displacement in the x-direction
dy	for constrained displacement in the y-direction
dz	for constrained displacement in the z-direction
rx	for constrained rotations about the x-axis
ry	for constrained rotations about the y-axis
rz	for constrained rotations about the z-axis

Remarks

The **mpc** command assigns constraints to be shared by a set of nodes. This set of nodes is defined by the **nset** command. The nodes in the set share a specified degree of freedom. The first node is the master node for those codes requiring a master node. When a merge command such as **t**, **tp**, or **stp** is issued, those nodes in a **mpc** are not merged together.

nr non-reflecting boundary

nr fset *face_set*

ol identifies a face of the mesh as an outlet for fluid flow

ol fset *face_set*

rigid create a rigid body from a nodal set

rigid *node_set_name options ;*

where

options can be

cid *local_sys_id*	
rgm *rigid_body_material_number*	for NIKE3D
com *x_center y_center z_center*	
trm *translational_mass*	

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

lcid *local_coordinate_system_for_inertia* for inertia tensor
ixx *xx-component_of_inertia_tensor*
ixy *xy-component_of_inertia_tensor*
ixz *xz-component_of_inertia_tensor*
iyy *yy-component_of_inertia_tensor*
iyz *yz-component_of_inertia_tensor*
izz *zz-component_of_inertia_tensor*
trv *x_velocity y_velocity z_velocity*
rtv *x_rot_velocity y_rot_velocity z_rot_velocity*

Remarks

This command allows you to append nodes to an existing rigid body for NIKE3D, or to create a nodal rigid body for LS-DYNA. The nodes appended to a NIKE3D rigid body inherit constraints, and so no constraints can be specified; the material number of the existing rigid body is required. For LS-DYNA, the nodal rigid body is created and all properties of the rigid body can be specified.

rml remove specific loads or conditions on a set of nodes

rml *nodes type*
where
nodes can be
n *node_number* a node number
rt *x y z* a Cartesian coordinate
cy *rho theta z* a cylindrical coordinate
sp *rho theta phi* a spherical coordinate
nset *name_of_set* a node set
type can be
all *load_curve_#* for all conditions for a load case (curve)
fc *load_#* for Nodal force by load curve
fd *load_curve_#* for Nodal displacement
fv *load_curve_#* for Nodal velocity
ft *load_curve_#* for Nodal velocity
tm for Initial temperature
sw *stone_wall_#* for Nodes impacting a stone wall
v for Electrostatic temperature
acc *load_curve_#* for Nodal acceleration
fa for Fixed nodal angular acceleration
mp for Magnetic potential

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

mom *load_#* for Nodal moment about an axis
lb *local_system_#* for Local coordinate nodal constraints
ve for Velocity
te for Temperature
si *interface_#* **S** for Nodal sliding interface
tepro *load_curve_# scale base_temp* for Temperature
where load_curve_#, stone_wall_#, interface_#, or local_system_#
can be 0 for all cases

Remarks

There is no undo command for the Merge Phase. However, you can remove a condition placed on a node set using the **rml** command. The **rml** command is more general than an **undo** because it can be used to remove conditions from any node set, whether or not the condition was applied directly to the node set in the first place.

rsl restore specific loads or conditions on a set of nodes

rsl *nodes type*
where
nodes can be
n *node_number* a node number
rt *x y z* a Cartesian coordinate
cy *rho theta z* a cylindrical coordinate
sp *rho theta phi* a spherical coordinate
nset *name_of_set* a node set
type can be
all *load_curve_#* for all conditions for a load case (curve)
fc *load_#* for Nodal force by load
fd *load_curve_#* for Nodal displacement
fv *load_curve_#* for Nodal velocity
ft *load_curve_#* for Nodal velocity
tm for Initial temperature
sw *stone_wall_#* for Nodes impacting a stone wall
v for Electrostatic temperature
acc *load_curve_#* for Nodal acceleration
fa for Fixed nodal angular acceleration
mp for Magnetic potential
mom *load_#* for Nodal moment about an axis

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

lb *local_system_#* for Local coordinate nodal constraints
ve for Velocity
te for Temperature
si *interface_#* **S** for Nodal sliding interface
tepro *load_curve_# scale base_temp* for Temperature
where load_curve_#, stone_wall_#, interface_#, or local_system_#
can be 0 for all cases

Remarks

This command is the inverse of **rml**.

spotweld interactive selection of spot welds

spotweld *(no arguments)*

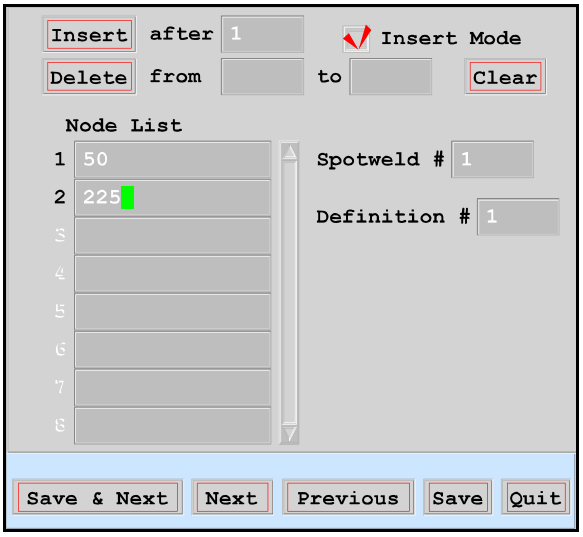


Figure 238 Spotweld dialogue box

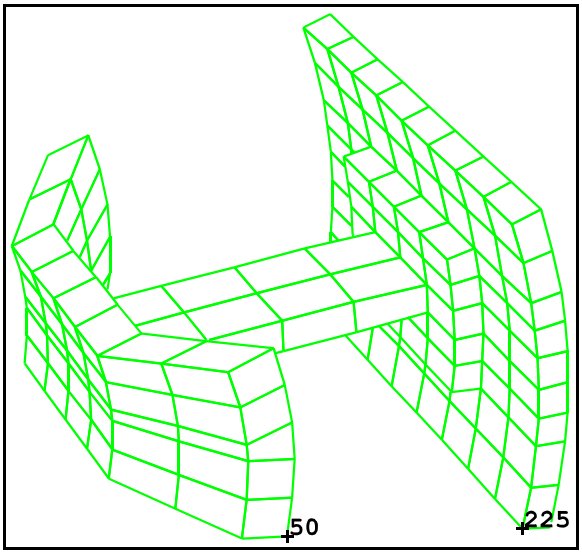


Figure 239 Selection of nodes with mouse

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

This feature is an easy and interactive way to create multiple spot welds using the mouse to click on the nodes. It is only available in the merge phase. It creates **spw** commands and writes them to the tsave (session) file.

The ***Spotweld #*** window should be set to the first spot weld number to be generated (see the **spw** command). Also select the appropriate spot weld property definition number (see the **spwd** command). Then click on nodes to be tied as a spotweld. Typically only two are needed, but NASTRAN allows for many nodes to be tied as a rigid body, so you can click on more than 2 nodes for NASTRAN. Then click on the ***Save & Next*** button. Repeat this for every spot weld.

An existing spot weld can be changed. Select the spot weld number by typing it into the ***Spotweld #*** window and entry. The node numbers can be changed. Deselect the ***Insert Mode*** check mark. Click on the node number in the Node List that you wish to change and then select a new node in the picture of the mesh in the physical window. Nodes can be deleted (use the ***Delete*** button) or a node can be inserted (use the ***Insert*** button) after a node already in the Node List. Be sure to ***Save*** this new spot weld.

You can scroll through the existing spot welds by clicking on the ***Next*** or ***Previous*** button. The nodes in the physical window will be marked.

Use the **co spw** feature to view each spot weld.

spw create a single spot weld

spw *id option nodes* ;

where

id	positive integer identifying the spot weld

where an *option* can be

spwd *spwd_id*	see **spwd** for the property number

where a *node* can be

ann *node_#*	which can be any literal node number
node *node_#*	which must be an existing node number
loc *x0 y0 z0 material_#*	to locate closest node
locc *x0 y0 z0 material_#*	to locate closest node and change its location

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

A literal node number will be output exactly as it is specified in this command. It does not have to refer to an existing node. This option makes it possible to spot weld between two separately generated meshes. Typically one or both meshes have node offsets (see **offset**).

This will generate the

*CONSTRAINED_SPOTWELD

and

*CONSTRAINED_SPOTWELD_FILTERED_FORCE

cards in LSDYNA. Use the **spwd** command to define the properties of the spot welds. When doing so, be sure to specify both the number of force vectors and the time window to activate the filtered force card.

For NASTRAN, the RBE2 card is generated.

Use the **co spw** feature to view each spot weld.

spwd spot weld property definitions

spwd *id options* ;

where

id	positive integer identifying the spot weld properties definition

where an *option* can be

dx	displacement in the x-direction for all nodes
dy	displacement in the y-direction for all nodes
dz	displacement in the z-direction for all nodes
rx	rotation about the local x-axis for all nodes
ry	rotation about the local y-axis for all nodes
rz	rotation about the local z-axis for all nodes
sn	nornal_force
ss	shear_force
n	normal_force_exponent
m	shear_force_exponent
tf	failure_time

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

ep	effective_plastic_strain
nf	#_force_vectors
tw	time_window

Remarks

This command defines the properties of a spotweld. These definitions are needed by the **spw** command and the associated interactive command **spotweld**. The shared degrees of freedom flags are used for NASTRAN. The other options are for LSDYNA.

spwf spot welds for LSDYNA material 100

spwf *points sw_material 1st_contact 2nd_contact*

where *points* can be

eqsp *3D_curve_# flag #_spotwelds*

where *flag* can be

0	to include the last point
1	to not include the last point

pnts *3D_curve_# flag*

where f*lag* can be

0	to include the last point
1	to not include the last point

rt *x y z*	Cartesian coordinates
cy *rho theta z*	cylindrical coordinates
sp *rho theta phi*	spherical coordinates

where *1st_contact* and *2nd_contact* can be

mat *material_number*

Remarks

Use the **co spwf** feature to view each spot weld.

sw select nodes that may impact a stone wall

sw *nodes stone_wall_#*

where

nodes can be:

n *node_number*	node number

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

rt *x y z*	a Cartesian coordinate
cy *rho theta z*	a cylindrical coordinate
sp *rho theta phi*	a spherical coordinate
nset *name_of_set*	a node set

Remarks

First use the **plane** command to define the stone wall.

syf assign faces to a numbered symmetry plane with failure

syf fset *face_set symmetry_plane_# failure*

trp create tracer particles

trp *tracking list_options ;*
where
tracking can be **fixed** or **free**, and
an *option* is followed by some parameters
time *start_time*
point *x0 y0 z0*
lnpy *x1 y1 z1 x2 y2 z2 #_tracers*

Remarks

This defines the location and properties of Tracer Particles in Ls-dyna.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

15. Radiation and Temperature Commands

These commands let you set various boundary conditions related to radiation and temperature. See also the radiation and temperature commands in the Part Phase.

bf bulk fluid

bf fset *face_set id_# load amplitude a b*

where

face_set	name of the face set
id_#	**bfd** bulk fluid identification number
load	load curve number
amplitude	multiplier of the load curve
a	exponent a
b	exponent b

Remarks

Use this command to identify those faces (surfaces) which are to be part of the bulk fluid (node) calculation.

Use the **orpt** command to orient the faces as desired.

Use the **co** command with the **bf** option to display the bulk fluid faces.

cv boundary convection

cv fset *face_set* $load_curve_1$ $amplitude_1$ $load_curve_2$ $amplitude_2$ *exponent*

where

face_set	name of the face set
$load_curve_1$	first load curve number or zero,
$amplitude_1$	amplitude factor for the first load curve,
$load_curve_2$	second load curve number or zero,
$amplitude_2$	amplitude factor for the second load curve, and
exponent	exponent.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

First use the **orpt** command to specify the surface orientation; that is, how to orient the normal vector.

A zero load curve number means that the condition is constant in time. If a curve is specified which has not been defined, a warning message will be issued.

cvt convection thermal loads

cvt fset *face_set coefficient temperature*

where

face_set	name of the face set
coefficient	film coefficient
temperature	temperature near convection

Remarks

First use the **orpt** command to specify the surface orientation; that is, how to orient the outward normal vector.

This command is used to create convection thermal loads for ANSYS. The first parameter is the film *coefficient*. This is followed by the *temperature* near convection. This command will create the EP cards for the ANSYS input file.

fl prescribed boundary flux

fl fset *face_set load_curve_# amplitude*

where

face_set	name of the face set
load_curve_#	a load curve number
amplitude	amplitude constant

Remarks

First use the **orpt** command to specify the surface orientation; that is, how to orient the outward normal vector.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

ft prescribed temperature

ft *nodes load_# temperature*

where

nodes can be

n *node_number*	for a node number
rt *x y z*	for a point in Cartesian coordinates
cy *rho theta z*	for a point in cylindrical coordinates
sp *rho theta phi*	for a point spherical coordinates
nset *name_of_set*	for a node set

Remarks

This specifies a time-dependent temperature boundary condition on the nodes. It is assumed that the temperature used by the appropriate simulation code will be the product of the temperature and the amplitude of the load curve at the appropriate time in the simulation.

rb prescribed radiation boundary condition

rb fset *face_set* $load_curve_1$ $amplitude_1$ $load_curve_2$ $amplitude_2$

where

face_set	name of the face set
$load_curve_1$	a load curve number
$amplitude_1$	amplitude constant
$load_curve_2$	a load curve number
$amplitude_2$	amplitude constant

Remarks

First use the **orpt** command to specify the surface orientation; that is, how to orient the outward normal vector.

If a load curve number is specified as zero, then the condition is constant in time.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

re radiation enclosure

re fset *face_set 0 temperature obstruction_flag*

where

face_set	name of the face set
temperature	constant
obstruction flag is:	
yes	to include surface obstruction calculations
no	to not include surface obstruction calculations

Remarks

First use the **orpt** command to specify the surface orientation; that is, how to orient the outward normal vector.

This command generates enclosure radiation data for TOPAZ3D. Use the **emissivity** and **rband** commands to specify the emissivity and wavelength breakpoint tables associated with the enclosure radiation data. For more details, see the TOPAZ3D manual.

te constant temperature for all nodes

These temperatures will override the temperatures specified by **temp**. When this command is invoked, all nodal temperatures are assumed to be specified. See the **temp** commands.

te *nodes load_# temperature*

where

nodes can be

n *node_number*	for a node number
rt *x y z*	for a point in Cartesian coordinates
cy *rho theta z*	for a point in cylindrical coordinates
sp *rho theta phi*	for a point spherical coordinates
nset *name_of_set*	for a node set

tepro nodal temperature profile

tepro *nodes load_# amp_expr ; temp_expr ;*

where

nodes can be

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

n *node_number* for a node number or zero
rt *x y z* for a point in Cartesian coordinates
cy *rho theta z* for a point in cylindrical coordinates
sp *rho theta phi* for a point spherical coordinates
nset *name_of_set* for a node set

amp_expr for a FORTRAN expression forming the scale factor
temp_expr for a FORTRAN expression forming the base temperature

Remarks

Both the base temperature and load curve scaling factor can be functions of the x, y, and z-coordinates of the node. This command is used for input to DYNA3D, LS-DYNA, and NIKE3D.

tm initial temperature

tm *nodes load_# temperature*
where
nodes can be
n *node_number* for a node number
rt *x y z* for a point in Cartesian coordinates
cy *rho theta z* for a point in cylindrical coordinates
sp *rho theta phi* for a point spherical coordinates
nset *name_of_set* for a node set

Remarks

This command does not invoke temperatures for all nodes like the **temp** commands and the **te** and **tei** commands.

vvhg volumetric heat generation w/ functional amplitude

vvhg *set_name load_# expression ;*
where
set_name element set name
load_# load curve number
expression amplitude expression

Remarks

The expression can be any valid FORTRAN like expression with x, y, and z. The coordinates used in this expression will be the centroid of the element.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

16. Electric Conditions

efl electric flux boundary condition

efl fset *face_set value_of_flux*

Remarks

This command produces four-node polygons with an assigned constant flux, one polygon for each face within the region.

mp constant magnetic potential

mp *nodes potential*

where

nodes can be

n *node_number*	for a node number
rt *x y z*	for a point in Cartesian coordinates
cy *rho theta z*	for a point in cylindrical coordinates
nset *name_of_set*	for a point spherical coordinates
sp *rho theta phi*	for a node set

v constant nodal electrostatic potential boundary condition

v *nodes potential*

where

nodes can be

n *node_numbe*r	for a node number
rt *x y z*	for a point in Cartesian coordinates
cy *rho theta z*	for a point in cylindrical coordinates
sp *rho theta phi*	for a point spherical coordinates
nset *name_of_set*	

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

17. Sets

These commands are unique to the Merge phase. Named sets are a useful tool in the definition of boundary conditions and properties in the mesh and form an alternative to selecting regions in the Part phase. An arbitrary selection can be made and this is the advantage to using set functions in the Merge phase. The disadvantage is that the selection may no longer be parametric. If you go back and make a change to the mesh, you may have to redefine the set since the element and node numbers have changed.

The name of the set can be up to 8 alphanumeric characters long. Each name of the set must be unique.

In some of the set commands, the logical or Boolean set operators AND and OR are used to create new sets from existing sets. The AND operator between two sets means to take their intersection. This should not be confused with the common usage of and which might be interpreted to mean the addition of two sets. The OR operator does this function.

adnset add nodes to an ordered node set

adnset *set_name insertion_node before/after list_of_nodes*

where

set_name	is the name of the node set
insertion_node	is the sequence number of the node for the insertion point
before/after	is either
0	for the insertion before the insertion node, or
1	for the insertion after the insertion node
list_of_nodes	is the list of nodes, which will be inserted to the ordered node set.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

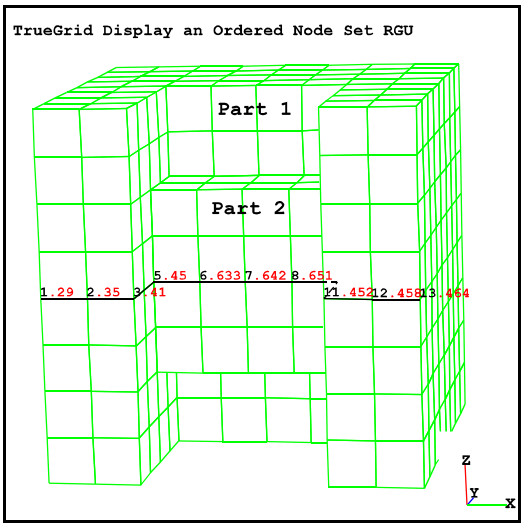


Figure 240 after **adnset**

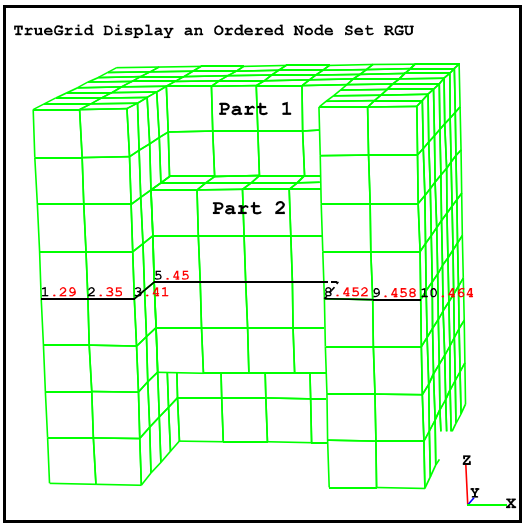


Figure 241 before **adnset**

This command adds or inserts a list of nodes into an ordered node set.

Example

Part 1 is created and a region is deleted. The node set named RGU is created using the index progression `1 2;-1;-3;`. The index progression `-2;1 2 ;-3;` is used to enhance the RGU node set by union. Part 2 is defined. Ordered node set RGU is labeled (**Figure 241**).

Then the nodes `633 642 651` are added to the ordered node set RGU, after sequence node number `5`. The result is displayed on **Figure 240**. The command file follows:

```
block 1 3 7 9; 1 3 5 7 9;1 3 5 7 9;1 3 7 9; 1 3 5 7 9;1 3 5 7 9;
dei 2 3;1 3;1 5;
nseti  1 2;-1;-3;= RGU
nseti -2;1 2 ;-3;OR RGU
block 1 5;1 3;1 3 5;3 7;3 5;3 5 7;
merge
dap
nset RGU OR L 456 454 452 458 464 ;
labels onset RGU 1 0
adnset RGU 5 1 633 642 651 ;
labels onset RGU 1 0
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

crvnset order a segment of an ordered node set (3D curve)

crvnset *set_name first_node last_node 3D_curve-#*

where

set_name	is the name of the node set
first_node	is the sequence number of the node
last_node	is the sequence number of the node
3D_curve-#	is the number of the 3D curve used for reordering of the nodes in the set.

Remarks

The nodal sequence number, referred to above, can be from 0 to the total number of nodes in the set. The sequence number **1** is always the first node in the set and sequence number **0** is the last node number in the set. Each node is projected to a 3D curve and it is assigned the arc length of the projected point on the 3D curve. Then the nodes are ordered according to the arc length. The effect is that the ordered set of nodes will approximate the shape and order of the 3D curve. This can be useful if the mesh is to be parameterized. Direction of the 3D curve plays an important role. If it is reversed, the ordering of the nodes in the set will be reversed. Use a negative curve number to accomplish this.

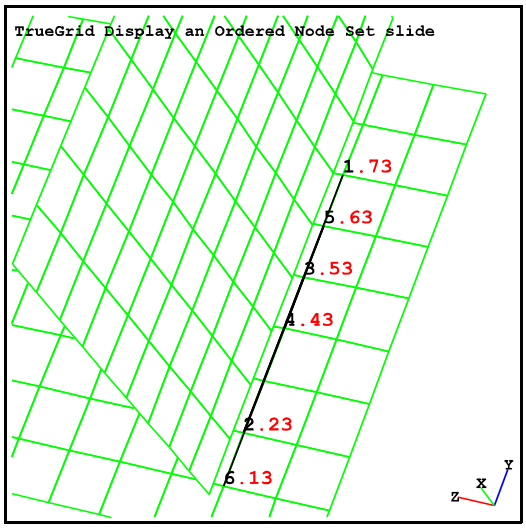


Figure 242 before **crvnset**

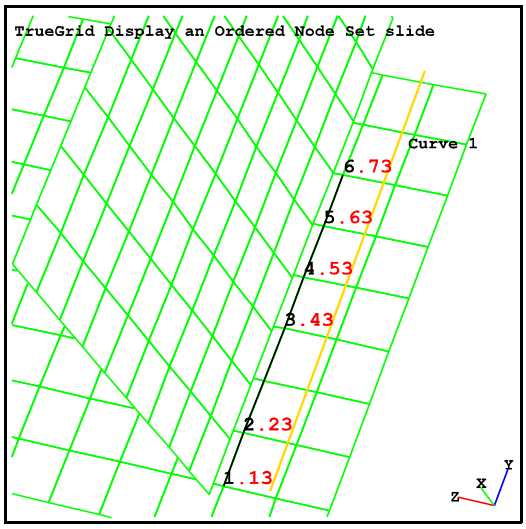


Figure 243 after **crvnset**

Example

The node set **slide** is at first defined by node numbers without any order (**Figure 242**). Then it is ordered according to curve 1 (**Figure 243**).

```
block -1;1 10;1 10;1;1 10;1 10;
block 1 10;1 10;-1;1 10;1.8 11.1;3.2;
merge
nset slide = L 73 23 53 43 63 13;
labels onset slide 1 0
curd 1 lp3 1.3 2 2 1.5 10 2;;
crvnset slide 1 0 1
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

eset add/remove elements to/from a set of elements

eset *set_name operator option*;

where

set_name is the name of the element set

operator can be

=	for initial assignment,
AND	for intersection with element set,
OR	for union with the element set,
-	for removal from the element set
RPL	to replace one element type (only with **LS**, **LB**, **LBM**, **QS**, and **QB**)

option can be

LBM *element_list* ;	linear beam element list
LS *element_list* ;	linear shell element list
QS *element_list* ;	quadratic shell element list
LB *element_list* ;	linear brick element list
QB *element_list* ;	quadratic brick element list
S set_name	a set of elements
MT *material*	elements of a material
SRF *surface tolerance #_nodes*	elements within a tolerance of a surface
CRV *curve tolerance #_nodes*	elements within a tolerance of a curve

Remarks

The initial assignment creates a face set. If the face set with the same name already existed, then it is deleted and recreated. The intersect operator redefines a face set to be only those faces which are found to be both in the original set and among the selected faces. Selected faces can be added by using the union operator. This causes any selected faces to be included in a set, if it is not already in that set. The minus operator removes all faces in a set which are among the selected faces.

The **SRF**, **CRV**, and **MT** options only consider those elements that are in the display set. If a part or material is removed from the picture, those elements will not be searched. **SRF** and **CRV** options have a number of nodes as a parameter which varies depending on the type of element. An element will be selected if the specified number of corner nodes of the element are within tolerance. 0 nodes means to use the element centroid.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

fset add/remove faces to/from a set of faces

fset *set_name operator faces*

where *set_name* is the name of the face set

where *operator* can be

=	for initial assignment
AND	for intersection with face set
OR	for union with the face set
-	for removal from the face set

where *faces* can be

srf *surface_id tolerance #_nodes*	specified number of nodes fall within the specified distance from the surface
set *set_name*	set operation with this previously defined face set
enumerations ;	each type of face must be in a separate list

where an *enumeration* can be

ls *list_elements* ;	list linear shell elements
lb1 *list_elements* ;	list linear bricks
lb2 *list_elements* ;	list linear bricks
lb3 *list_elements* ;	list linear bricks
lb4 *list_elements* ;	list linear bricks
lb5 *list_elements* ;	list linear bricks
lb6 *list_elements* ;	list linear bricks
qs *list_elements* ;	quadratic shell elements
qb1 *list_elements* ;	list quadratic bricks
qb2 *list_elements* ;	list quadratic bricks
qb3 *list_elements* ;	list quadratic bricks
qb4 *list_elements* ;	list quadratic bricks
qb5 *list_elements* ;	list quadratic bricks
qb6 *list_elements* ;	list quadratic bricks

Remarks

The initial assignment creates a face set. If the face set with the same name already existed, then it is deleted and recreated. The intersect operator redefines a face set to be only those faces which are found to be both in the original set and among the selected faces. Selected faces can be added by using the union operator. This causes any selected faces to be included in a set, if it is not already in that set. The minus operator removes all faces in a set which are among the selected faces. See also pg.258 for labeling of the face sets.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

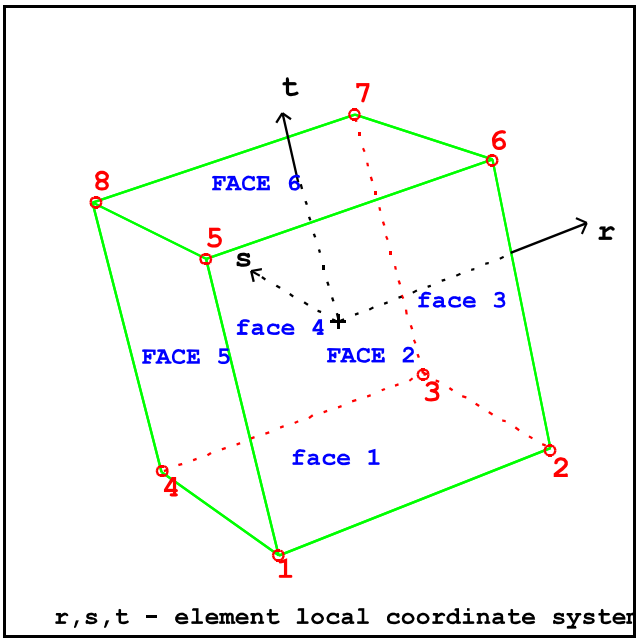


Figure 244 face order in brick element

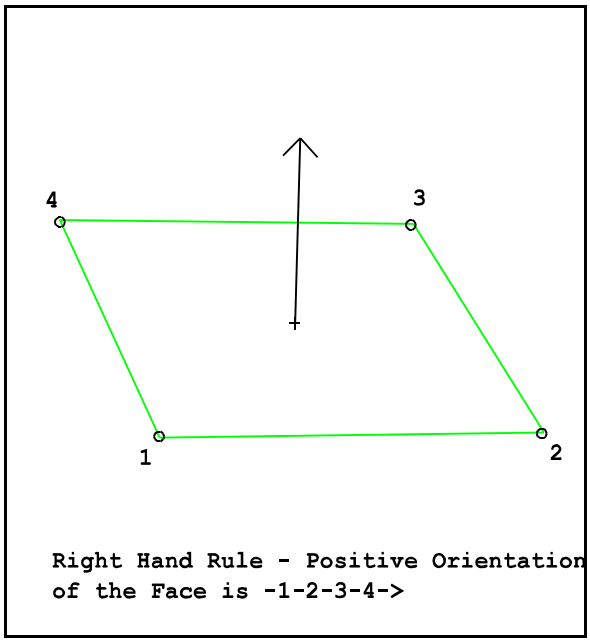


Figure 245 orientation of nodes in face

Faces in face sets are identified by an element number and an order number of a face in the element. Face order in the element is shown in 244. Nodes in the face are ordered by the right hand rule The vector in 245 is always oriented outward from the element.

The **SRF** option only considers those elements that are in the display set. If a part or material is removed from the picture, those elements will not be searched. The **SRF** option has a number of nodes as a parameter which varies depending on the type of element. An element will be selected if the specified number of corner nodes of the element are within tolerance. 0 nodes means to use the element centroid.

infol information on nodal loads

infol *nodes type*

where

nodes can be

N *node_number*	node number
RT *x y z*	node closest to Cartesian point
CY *rho theta z*	node closest to cylindrical point
SP *rho theta phi*	node closest to spherical point
NSET *name_of_set*	node set by name

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

type can be

ALL *load_curve_#*	any type
FC *load_curve_#*	forces
FD *load_curve_#*	fixed displacements
FV *load_curve_#*	fixed velocity
FT *load_curve_#*	forced temperature
TM	initial temperature
SW *stone_wall_#*	stone wall
V	electrostatic potential
ACC *load_curve_#*	acceleration
FA	fixed angular displacement
MP	magnetic potential
MOM *load_curve_#*	moment
LB *local_coordinate_system_#*	constraint in local coordinate system
VE	velocity
TE	temperature
SI *sliding_interface_#*	nodal sliding interface
NPB	nodal print blocks
TEPRO *load_curve_#*	temperature profile

where

load_curve_#, *stone_wall_#*, *local_coordinate_system_#* can be **0** for all cases.

Remarks

Command **infol** prints information about nodes with a specific load or condition. The results on the **infol** are printed to the dialog window and to the session file. See the **rml** command for example of use.

mvnset move a subset of nodes in an ordered node set

mvnset *set_name first_node last_node insertion_node before/after*

where

set_name	is the name of the node set
first_node	is the first node sequence number of the group of nodes to be moved
last_node	is the last node sequence number of the group of nodes to be moved
insertion_node	is the sequence number of the node for the insertion point

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

before/after	is either
0	for before the insertion node
1	for after the insertion node

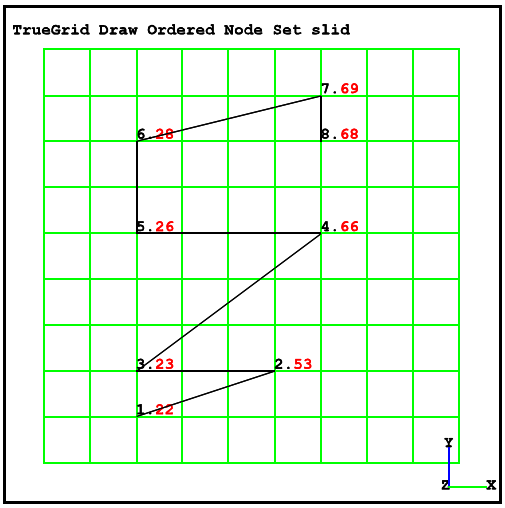


Figure 246 before **mvnset**

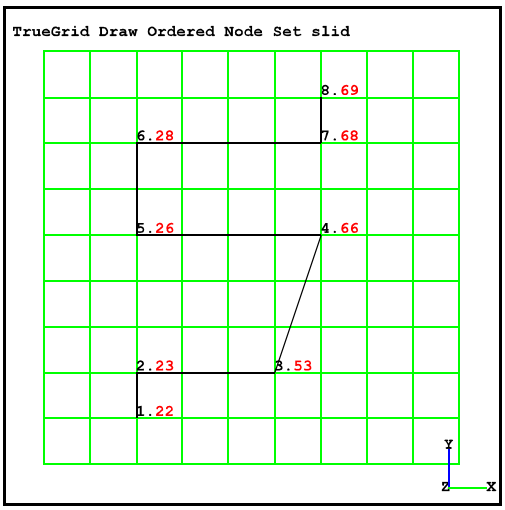


Figure 247 after **mvnset**

Remarks

Command **mvnset** moves a subset of nodes in an ordered nodal set. This can be used with **onset** to get the proper ordering of nodes. If the first sequence number is greater than the second sequence number, then the interval of nodes will have their order reversed before they are inserted back into the ordered set.

Example

The node set SLID is initialized by the list of nodes (**Figure 246**). The node set is reordered by **mvnset** command (**Figure 247**).

```
block 1 10;1 10;-1;1 10;1 10;0;
merge
nset SLID = L 22 53 23 66 26 28 69 68 ;
labels onset SLID 1 0
mvnset SLID 2 2 3 1
mvnset SLID 7 7 8 1
labels onset SLID 1 0
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

nset add/remove nodes to/from a set of nodes

nset *set_name operator node_selection*

where *set_name* is the name of the node set

where *operator* can be

=	for initial assignment
AND	for intersection with node set
OR	for union with the node set
+	to append the selected nodes to the node set
-	for removal from the node set

where *node_selection* can be

L *list;*	list of node numbers
S *set_name*	named set of nodes
SRF *surface_number tolerance*	nodes near a surface
CRV *curve_number tolerance*	nodes near a curve
MT *material_number*	
LOAD *load*	nodes with a specific load

where *load* can be

ALL *load_curve_#*	any type
FC *load_curve_#*	forces
FD *load_curve_#*	fixed displacements
FV *load_curve_#*	fixed velocity
FT *load_curve_#*	forced temperature
TM	initial temperature
SW *stone_wall_#*	stone wall
V	electrostatic potential
ACC *load_curve_#*	acceleration
FA	fixed angular displacement
MP	magnetic potential
MOM *load_curve_#*	moment
LB *local_coordinate_system_#*	constraint in local coordinate system
VE	velocity
TE	temperature
SI *sliding_interface_#*	nodal sliding Interface
NPB	nodal Print blocks
TEPRO *load_curve_#*	temperature Profile
B *constraint flag*	

where *constraint* can be

dx	constrain the x translation
dy	constrain the y translation

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

dz	constrain the z translation
rx	constrain the rotation about the x axis
ry	constrain the rotation about the y axis
rz	constrain the rotation about the z axis

where *flag* can be

1	constrained
0	unconstrained

Remarks

This command is probably the easiest and most versatile method for constructing a node set. The initial assignment creates a node set. If the node set with the same name already existed, then it is deleted and recreated. The intersect operator redefines a node set to be only those nodes which are found to be both in the original set and among the selected nodes. Selected nodes can be added by using the union operator. This causes any selected nodes to be included in a set, if it is not already in that set. The add operator will always append selected nodes to a set. This is used to create ordered node sets where duplicate nodes are allowed. The minus operator removes all nodes in a set which are among the selected nodes.

Example

The node set is defined by the list of node numbers in an arbitrary order (**Figure 248**).

```
cylinder 1 -3;1 -5 -7 -9 -13;1 -5
-9 -13 17;
          2.5 3;-45 -15 0 15 45;1
          3 5 7 9;
dei -2; 2 4; 2 3;
dei 1 2; -3; 2 3;
merge
nset xx = L
 44 45 81 96 97 104 105 132 147
148 219 220 316 329
 330 339:341 351 352 385 386;
labels nodeset xx
```

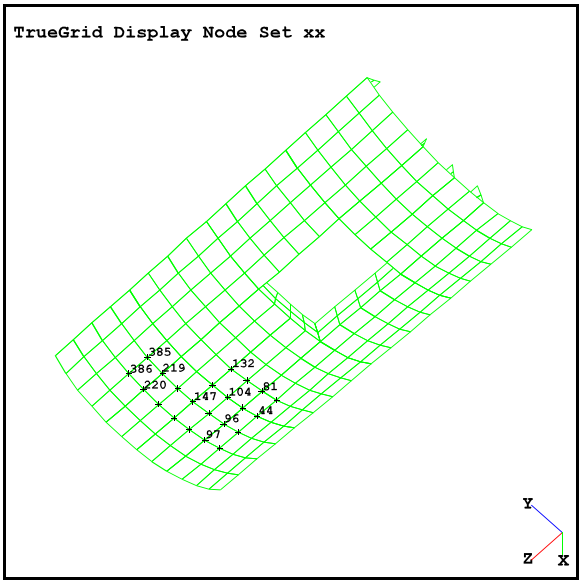


Figure 248 Node Set by **nset**

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

This example demonstrates the parametric capabilities of node sets. The mesh and the curve (circle) are shown (**Figure 249**). Node set Press_1 is defined by a distance .3 from curve number 1 (**Figure 250**). Node set Press_2 is defined by a distance of 1 from curve number 1 (**Figure 251**).

```
block 1 50;1 50;-1;0 10;0 10; 0;
curd 1 arc3 whole rt 7 7 0
 rt 6 6 0 rt 4 6 0; merge
nset Press_1 = crv 1 .3
labels nodeset  Press_1
nset Press_2 = crv 1 1
labels nodeset  Press_2
```

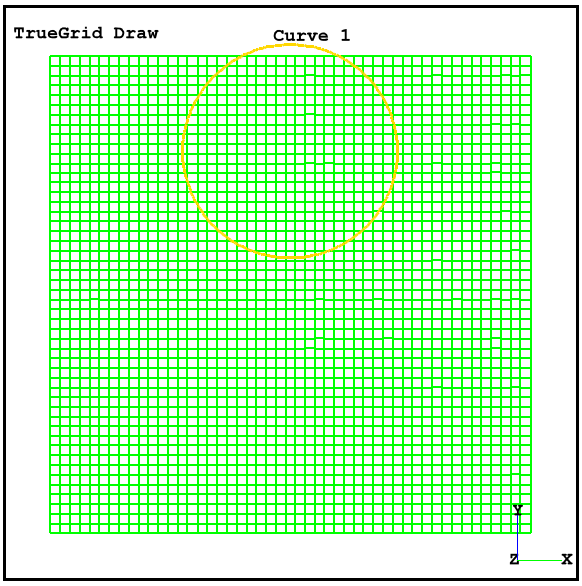


Figure 249 Mesh and Curve

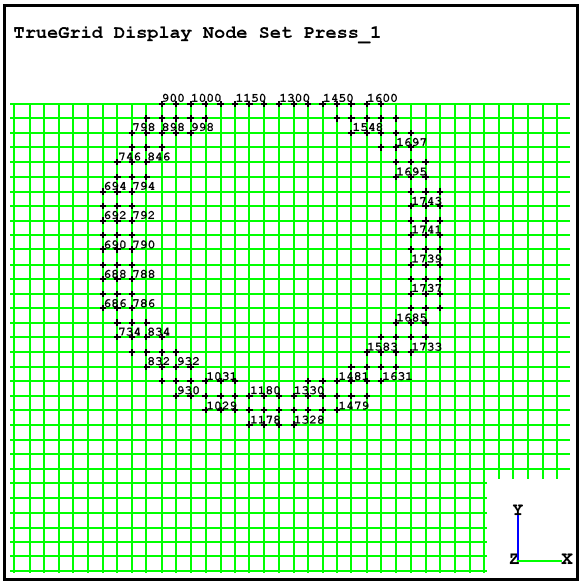


Figure 250 Node Set Press_1

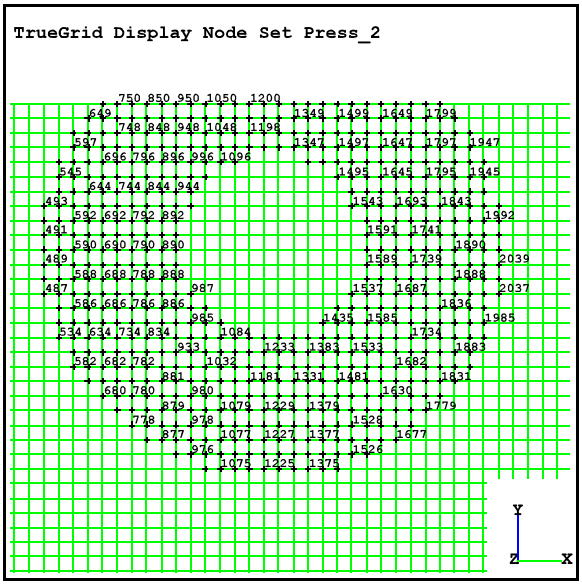


Figure 251 Node Set Press_2

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

Another useful application of the **nset** command is the definition of a sliding interface (**Figure 252**). The node set RAD1 is composed of all nodes within a distance of 2 units from the cylinder (**Figure 252**). The result of the **nset** command is on the **Figure 253**.

```
nset RAD1 = srf 6 2
labels nodeset RAD1
```

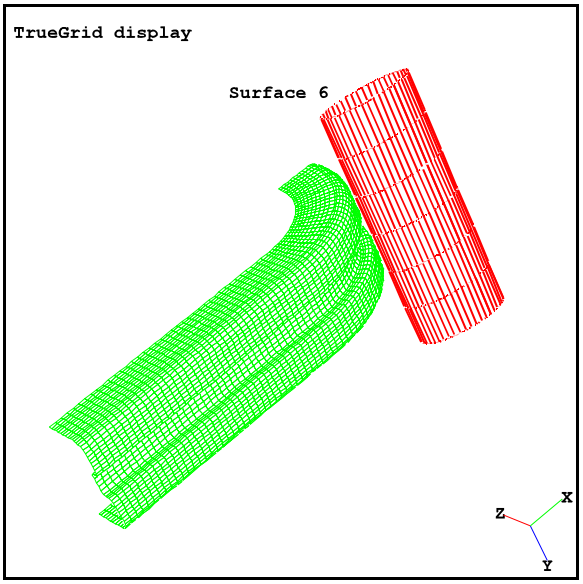


Figure 252 Cylinder and mesh

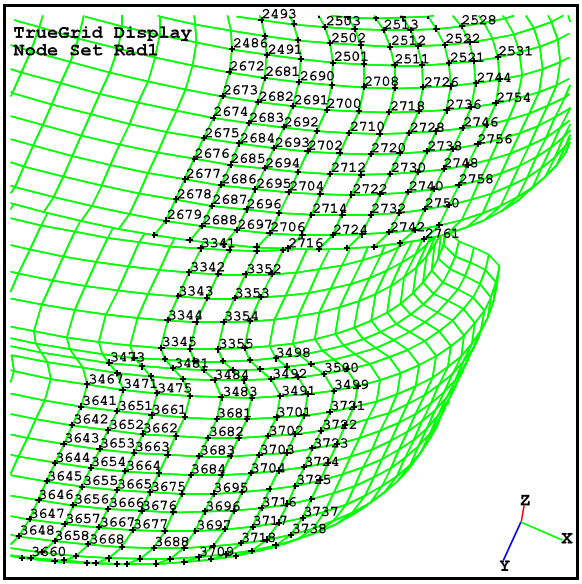


Figure 253 Node Set Rad1

onset order a segment of a nodal set

onset *set_name first_node last_node*

where

set_name	name of the node set
first_node	first node sequence number of the group of nodes to be moved
last_node	last node sequence number of the group of nodes to be moved.

Remarks

This command is useful when building a 1D slide line, for example, where the nodes must be in a certain order. The nodes can be included in the node set as each part in generated using the **nset**

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

command. Additional nodes can be included in a set using the **nset** command in the Merge Phase. Use the **labels** command under graphics to view the order of the nodes in the set. Use this command to reorder a segment of the nodes. Select the beginning and ending nodes of the sequence to be reordered. Use the sequence numbers which are in white. See also **mvnset** command.

pset create or modify a polygon set

pset *set_name operator selection*

where *set_name* is the name of the polygon set

where *operator* can be

=	for initial assignment
AND	for intersection with node set
OR	for union with the node set
+	to append the selected nodes to the node set
-	for removal from the node set

where the *selection* can be

l s_1 p_1 s_2 p_2 ... ;	for a list of polygons
where	
s_i	surface number
p_i	polygon number of that surface
s *polygon_set_name*	for another polygon set

Remarks

Associated with this command is an interactive feature to select or modify the polygons in a set with the ***Sets*** window from the Environment Window.

Polygon sets can then be turned into surfaces using (see the **sd** command):

sd *surface#* **poly** *polygon_set trans* ;

These features along with the **wrsd** can be used to sort out complex polygon surfaces and split them into multiple surfaces or remove features. This can also be used to create an normal offset by using the normal option as one of the transformation primitives.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

rml remove specific loads or conditions on a set of nodes

rml *nodes type*

where

nodes can be

N *node_number*	node number
RT *x y z*	node closest to Cartesian point
CY *rho theta z*	node closest to cylindrical point
SP *rho theta phi*	node closest to spherical point
NSET *name_of_set*	node set by name

type can be

ALL *load_curve_#*	any type
FC *load_curve_#*	forces
FD *load_curve_#*	fixed displacements
FV *load_curve_#*	fixed velocity
FT *load_curve_#*	forced temperature
TM	initial temperature
SW *stone_wall_#*	stone wall
V	electrostatic potential
ACC *load_curve_#*	acceleration
FA	fixed angular displacement
MP	magnetic potential
MOM *load_curve_#*	moment
LB *local_coordinate_system_#*	donstraint in local coordinate system
VE	velocity
TE	temperature
SI *sliding_interface_#*	nodal sliding interface
NPB	nodal print blocks
TEPRO *load_curve_#*	temperature profile

and where

load_curve_#, *stone_wall_#*, *local_coordinate_system_#* can be **0** for all cases

Remarks

The **rml** command is very useful in conjunction with the **readmesh** command to alter boundary conditions. The **rml** can be used also to edit the forces. To change of boundary condition, first remove it and them issue a new boundary condition.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
block 1 20;1 20; -1;0 20;0 20;0;        c mesh definition
merge
nset force = l 155 154 153 173 174 175 ;
                                 c node set force initialization
lcd 1 0 1 1 1;                   c load curve definition
fc nset force 1 1 1 0 0          c concentrated force definition
                                 c for the node set force
infol nset force all 1
     c information print for the node set force
     c for all conditions (stat is either on or off):
                        LOAD TABLE
NODE STAT  LOAD   CASE     X          Y          Z       AMPLITUDE
155  on    fc       1     1.0        0.0        0.0         1.0
154  on    fc       1     1.0        0.0        0.0         1.0
153  on    fc       1     1.0        0.0        0.0         1.0
173  on    fc       1     1.0        0.0        0.0         1.0
174  on    fc       1     1.0        0.0        0.0         1.0
175  on    fc       1     1.0        0.0        0.0         1.0
mom nset force 1 2 x             c moment around x definition
                                 c for the node set force
infol nset force all 1
     c information print for the node set force
     c for all conditions  (stat is either on or off):
                        LOAD TABLE
NODE STAT  LOAD   CASE    X          Y          Z       AMPLITUDE
 155  on    fc       1    1.0        0.0        0.0         1.0
 154  on    fc       1    1.0        0.0        0.0         1.0
 153  on    fc       1    1.0        0.0        0.0         1.0
 173  on    fc       1    1.0        0.0        0.0         1.0
 174  on    fc       1    1.0        0.0        0.0         1.0
 175  on    fc       1    1.0        0.0        0.0         1.0
 155  on    mom      1     1                               2.0
 154  on    mom      1     1                               2.0
 153  on    mom      1     1                               2.0
 173  on    mom      1     1                               2.0
 174  on    mom      1     1                               2.0
 175  on    mom      1     1                               2.0
rml nset force fc 1              c removal of the concentrated
               c forces of the node set force from the model
infol nset force all 1
     c information print for the node set force
     c for all conditions  (stat is either on or off):
                        LOAD TABLE
NODE STAT  LOAD   CASE     X          Y          Z       AMPLITUDE
155  off   fc       1     1.0        0.0        0.0         1.0
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
154  off   fc       1     1.0        0.0        0.0        1.0
153  off   fc       1     1.0        0.0        0.0        1.0
173  off   fc       1     1.0        0.0        0.0        1.0
174  off   fc       1     1.0        0.0        0.0        1.0
175  off   fc       1     1.0        0.0        0.0        1.0
155  on    mom      1     1                                2.0
154  on    mom      1     1                                2.0
153  on    mom      1     1                                2.0
173  on    mom      1     1                                2.0
174  on    mom      1     1                                2.0
175  on    mom      1     1                                2.0
```

rsl restore specific loads or conditions on a set of nodes

rsl *nodes type*

where

nodes can be

N *node_number*	node number
RT *x y z*	node closest to Cartesian point
CY *rho theta z*	node closest to cylindrical point
SP *rho theta phi*	node closest to spherical point
NSET *name_of_set*	node set by name

type can be

ALL *load_curve_#*	any type
FC *load_curve_#*	forces
FD *load_curve_#*	fixed displacements
FV *load_curve_#*	fixed velocity
FT *load_curve_#*	forced temperature
TM	Initial temperature
SW *stone_wall_#*	stone wall
V	electrostatic potential
ACC *load_curve_#*	acceleration
FA	fixed angular displacement
MP	magnetic potential
MOM *load_curve_#*	moment
LB *local_coordinate_system_#*	constrain in local coordinate system
VE	velocity
TE	temperature
SI *sliding_interface_#*	nodal sliding interface
NPB	nodal print blocks

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

TEPRO *load_curve_#* temperature profile

and where

load_curve_#, *stone_wall_#*, *local_coordinate_system_#* can be **0** for all cases

Remarks

The **rsl** command is the complement to the **rml** command. It can be used only after **rml** is issued. You can use the **infol** command to make sure which loads are turned off.

Example

```
infol nset force all 1
     c information print for the node set force
     c for all conditions  (stat is either on or off):
                         LOAD TABLE
NODE STAT  LOAD   CASE     X          Y          Z      AMPLITUDE
155  off   fc       1     1.0        0.0        0.0        1.0
154  off   fc       1     1.0        0.0        0.0        1.0
153  off   fc       1     1.0        0.0        0.0        1.0
173  off   fc       1     1.0        0.0        0.0        1.0
174  off   fc       1     1.0        0.0        0.0        1.0
175  off   fc       1     1.0        0.0        0.0        1.0
155  on    mom      1     1                                2.0
154  on    mom      1     1                                2.0
153  on    mom      1     1                                2.0
173  on    mom      1     1                                2.0
174  on    mom      1     1                                2.0
175  on    mom      1     1                                2.0
rsl nset force fc 1  c restore load case 1for the node set force
infol nset force all 1
     c information print for the node set force
     c for all conditions  (stat is either on or off):
                         LOAD TABLE
NODE STAT  LOAD   CASE     X          Y          Z      AMPLITUDE
155  on    fc       1     1.0        0.0        0.0        1.0
154  on    fc       1     1.0        0.0        0.0        1.0
153  on    fc       1     1.0        0.0        0.0        1.0
173  on    fc       1     1.0        0.0        0.0        1.0
174  on    fc       1     1.0        0.0        0.0        1.0
175  on    fc       1     1.0        0.0        0.0        1.0
155  on    mom      1     1                                2.0
154  on    mom      1     1                                2.0
153  on    mom      1     1                                2.0
173  on    mom      1     1                                2.0
174  on    mom      1     1                                2.0
175  on    mom      1     1                                2.0
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

rvnset remove a node subset

rvnset *set_name first_node last_node*

where

set_name	name of the node set
first_node	first node sequence number of the group of nodes to be moved
last_node	last node sequence number of the group of nodes to be moved.

Example

The node set SLID is defined, reordered and displayed (**Figure 254**). The nodes with sequence numbers `2,3,4,5` are removed from the node set SLID. The node set SLID is displayed in **Figure 255**.

```
labels onset SLID 1 0
c display labeled ordered set SLID
rvnset SLID 2 5
c remove node sequence position
c from 2 to 5 from the set SLID
labels onset SLID 1 0
c display labeled ordered set SLID
```

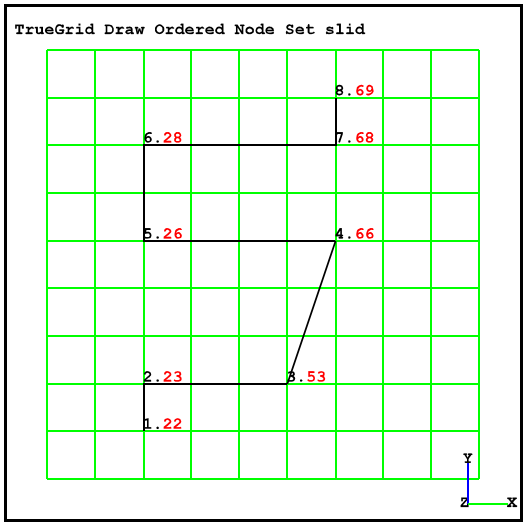


Figure 254 before **rvnset**

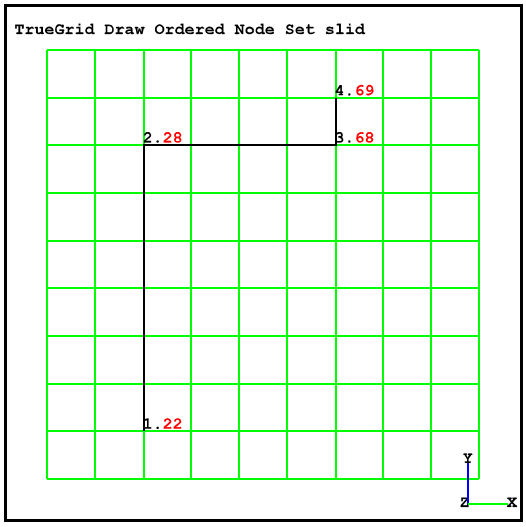


Figure 255 after **rvnset**

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

III. Global Commands

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

1. Materials

delmats delete a material definition

delmats *list* ;

where

list a list of material numbers of material models to be removed from the data base

Remarks

This command is useful when reading a model using the **readmesh** command. There may be material models for one FEM code which will not be used when the model is translated to a new format. Deleting the material models is the safest way to avoid complications.

2. Parts

There are several ways to create a part. The **block** command is recommended for most problems. Only rarely is a **cylinder** part justify because of its unusual behavior near the local z-axis. The **readmesh** creates several parts by reading a mesh file with a specific format. Each material within the mesh file becomes a part. The **blude** command is even more rarely used and was designed to build a fluid mesh around an existing shell mesh. The **beam** and **cbeam** commands are included to stay compatible with old versions of Ingrid, **True*Grid*®**'s predecessor. The **bm** command in the merge phase is the preferred method for generating a string of beams.

If the **linear**, **quadratic** or **partmode** command precedes the **block**, **cylinder**, or **blude** commands, it affects the meaning of those commands. These three commands cannot be issued within the scope of a part command. For that reason, they are no discussed in this section. They can be found under assembly commands.

There are commands to display selected parts of your model. When you enter the merge phase, all parts for graphics are automatically activated. When you enter into the part phase, only that part will be active in the graphics. When a sequence of parts is needed, you only need to give the first and last numbers separated by a colon.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

block create a brick-shaped part

block *i_indices ; j_indices ; k_indices ; x_coordinates y_coordinates z_coordinates*

where

i_indices, *j_indices*, and *k_indices* are lists of indices (not reduced indices). Each list of indices takes the form i1 i2 ... in, where i1=1 and |i1| < |i2| < ... < |in|. Normally each of these indices is a positive integer. But a negative integer can be used to create a shell surface, and a zero is used to separate regions of the mesh.

x_coordinates, *y_coordinates*, and *z_coordinates* are lists of values of physical coordinates, one for each index value: an x coordinate for each i index, a y coordinate for each j index, and a z coordinate for each k index. These define the physical locations of every region interface.

Example

The block part is defined by a list of i-indices and x-coordinates, a list of j-indices and y-coordinates, and a list of k-indices and z-coordinates. An example command file follows:

```
c Block part definition.
c  Indices in the x-direction (i-dir.) are:1 5 7 8 11.
c    x-coordinates are:-1.3 5.2 8 8.7 11.3 units
c  Indices in the y-direction (j-dir.) are: 1 3 5.
c    y-coordinates are:1.2 3.4 5.2 units.
c  Indices in the z direction (k-dir.) are:1 10.
c    Z-coordinates are:-5 5 units.

block 1 5 7 8 11;   1 3 5;   1 10;
      -1.3  5.2  8  8.7  11.3;
        1.2  3.4  5.2;
        -5  5;
```

This block is illustrated in the following 6 pictures. The faces that can be selected using the index bars correspond to the indices in the index lists. These faces partition the part into blocks and are sometimes referred to as i-, j-, and k-partitions for descriptive purposes.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

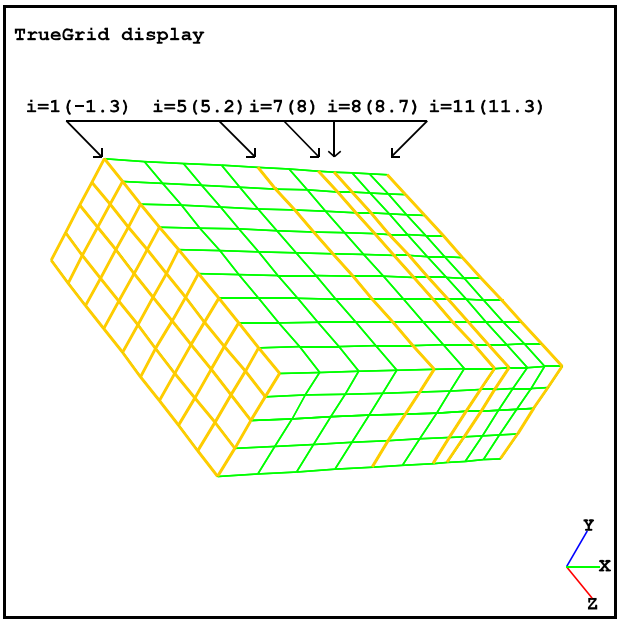


Figure 256 i-partitions and x-coordinates

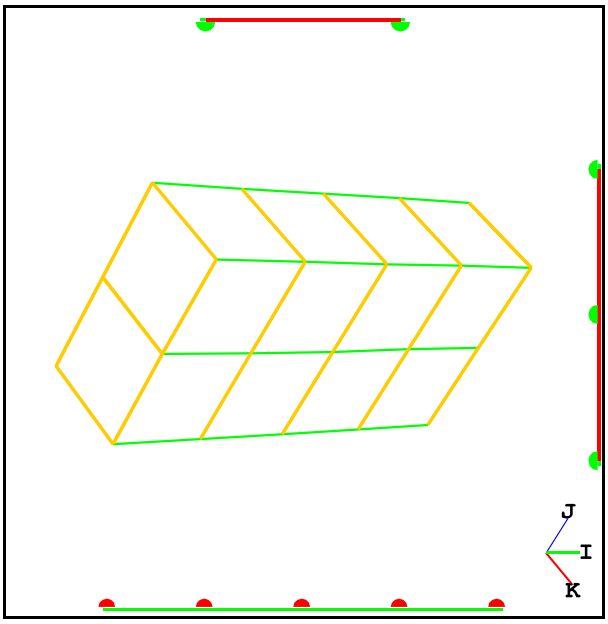


Figure 257 i-partitions

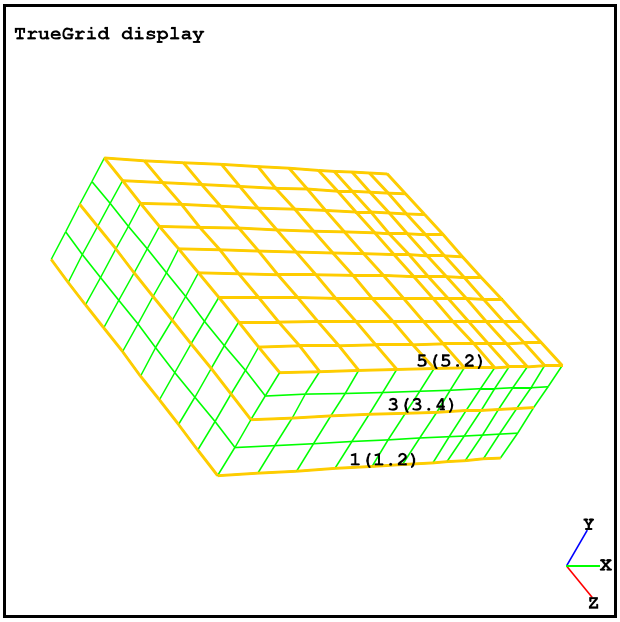


Figure 258 j-partitions and y-coordinates

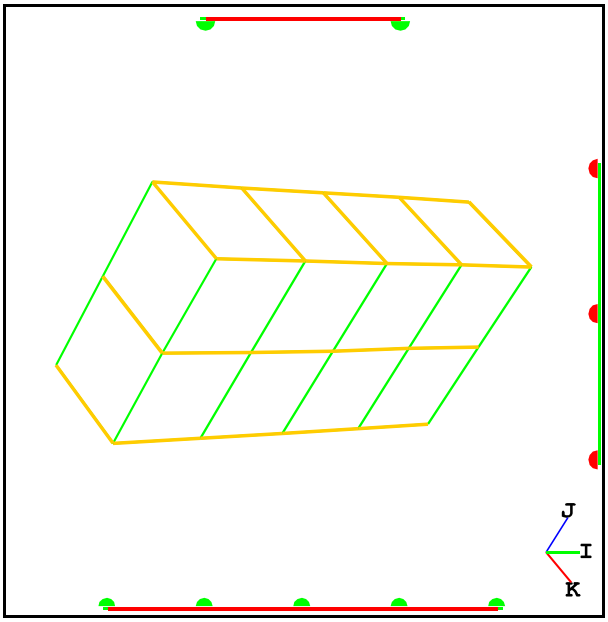


Figure 259 j-partitions

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

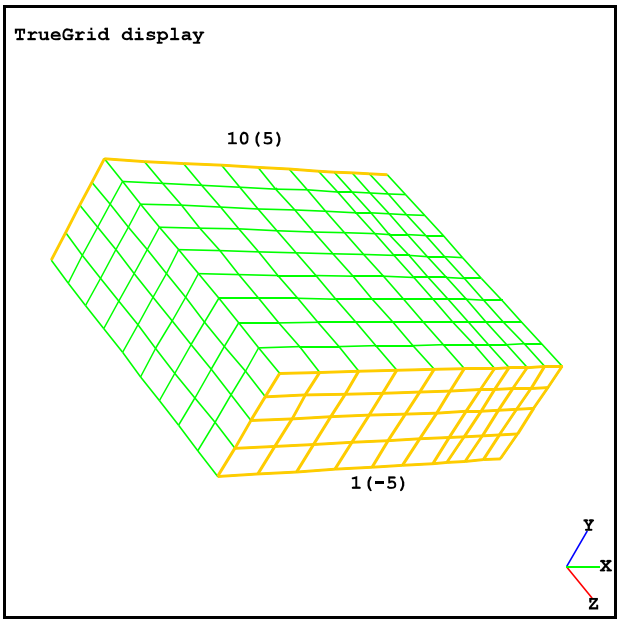


Figure 260 k-partitions and z-coordinates

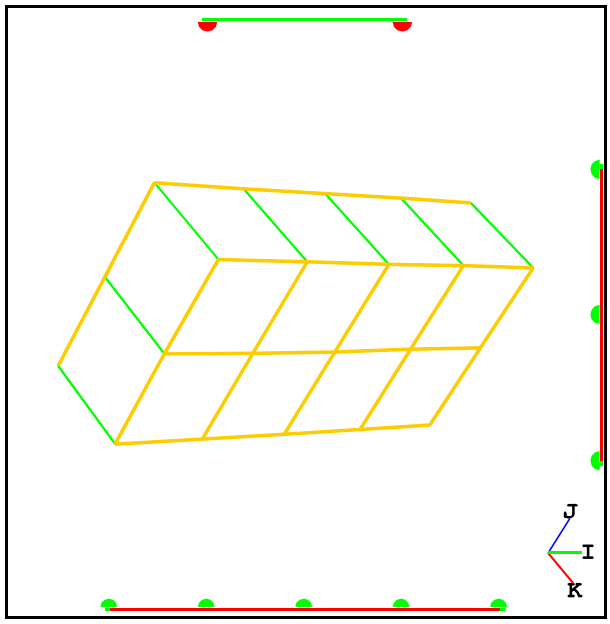


Figure 261 k-partitions

Remarks

This command is also discussed in the Introduction.

This command transitions the code to the Part Phase. This is the standard way to generate parts. In order to complete the Part Phase and return to the Control Phase, use the **endpart** or **control** command. In order to complete the Part Phase and return to the Merge Phase, use the **merge** command. In order to abort the part and return to the Control Phase, use the **abort** command.

When this command is issued, the previous part (if any) is ended as if the **endpart** command had been issued.

Six lists of numbers follow the **block** command. The first three lists consist of integers, each list terminated with a semi-colon. The second three lists consist of real numbers, each list optionally terminated by a semi-colon. The first list of integers must start with a 1 or -1. The integers that follow must be zero or have absolute value greater than the absolute values of the integers that preceded it in that list. These are the number of nodes to create in the first dimension of the computational mesh. A positive integer indicates that there will be a partition at that nodal index in the first dimension of the computational mesh, which can be referenced by most commands in the Part Phase. These partitions are used to break the part into multiple structured blocks. Since most

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

part commands reference only the partition sequence numbers and not the nodal indices, it is a trivial matter to change the density of the mesh by only changing this list of integers. Moreover, one may use parameter values here in order to be able to later alter the mesh density. The partition sequence numbers corresponding to the integers in this list are symbolic references to the structure of the mesh. When positive integers are used, solid elements are created. If you are generating solid elements exclusively, you may prefer the alternative method of specifying the number of elements in each list by first issuing the **partmode** command. A negative integer in the list also produces a partition in the mesh with a nodal index corresponding to the absolute value of the integer, with shell elements created along that partition in the computational mesh. A zero in the list means there is a gap in the mesh, i.e. the part is not connected. The second and third lists have the same meaning applied to the second and third dimensions in the computational mesh, respectively. The fourth list is a list of x-coordinates, one coordinate for each integer in the first list. This initializes each partition in the first dimension of the computational mesh. The fifth list consists of y-coordinates corresponding in like manner to the second list of partitions in the second dimension of the computational mesh. The sixth list consists of z-coordinates, corresponding in like manner to the third list of partitions in the third dimension of the computational mesh.

cylinder create a cylindrical part

cylinder *i_indices ; j_indices ; k_indices ; r_coordinates θ_coordinates z_coordinates*

where

i_indices, *j_indices*, and *k_indices*

are lists of indices (not reduced indices). Each list of indices takes the form i1 i2 ... in, where i1=1 and |i1| < |i2| < ... < |in|. Normally each of these indices is a positive integer. But a negative integer can be used to create a shell surface, and a zero can be used to separate parts of the mesh.

r_coordinates, *θ_coordinates*, and *z_coordinates*

are lists of values of physical coordinates, one for each index value: an r coordinate for each i index, a θ coordinate for each j index, and a z coordinate for each k index. These define the physical locations of every region interface.

Example

The cylinder part is defined by a list of i-indices and radii, a list of j-indices and angles, and a list of k-indices and z-coordinates. A sample command file follows:

```
c Cylinder part definition.
c  Indices in the radial direction (i-dir.) are:1 3 5 7 11.
c    Radii are:6 9 12 15 18 units
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
c  Indices in the angular direction (j-dir.) are:
c                                  1 5 9 13 17 21 25 29 33 37.
c    Angles are:0 50 80 120 160 200 240 280 310 360 degrees.
c  Indices in the z direction (k-dir.) are:1 3 5 7 9.
c    Z-coordinates are:-10 0 10 20 30 units.

cylinder 1 3 5 7 11; 1 5 9 13 17 21 25 29 33 37; 1 3 5 7 9;
          6 9 12 15 18;
           0 50 80 120 160 200 240 280 310 360;
            -10 0 10 20 30;
```

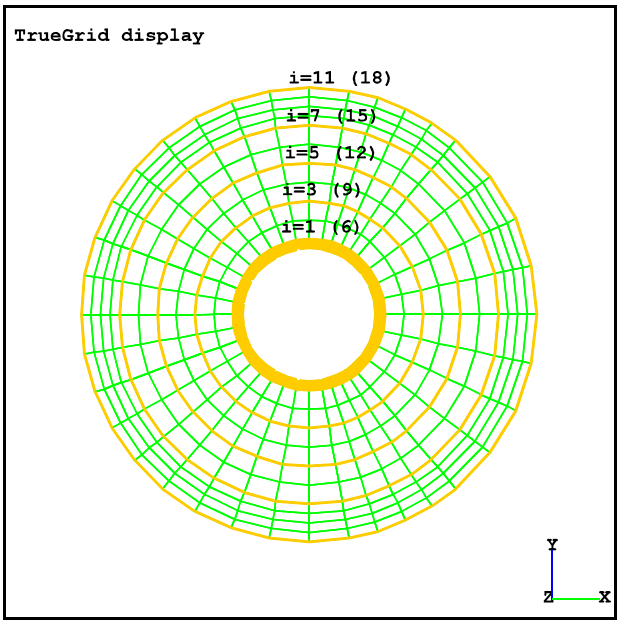


Figure 262 i-partitions and radiuses

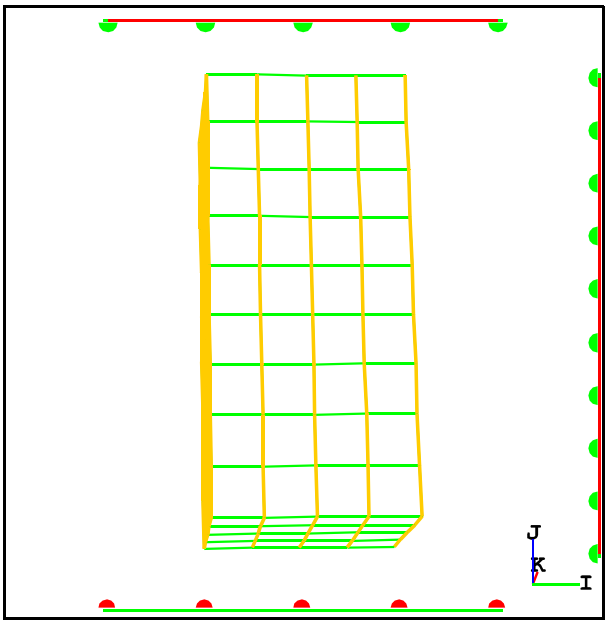


Figure 263 i-partitions

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

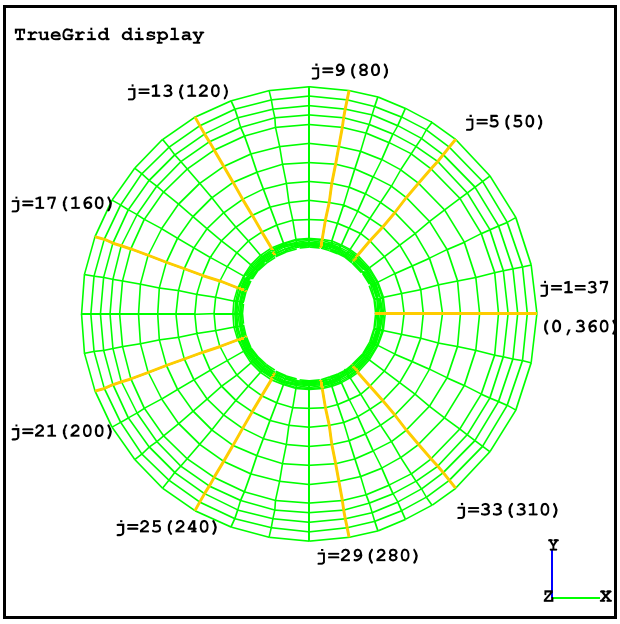


Figure 264 j-partitions and angles

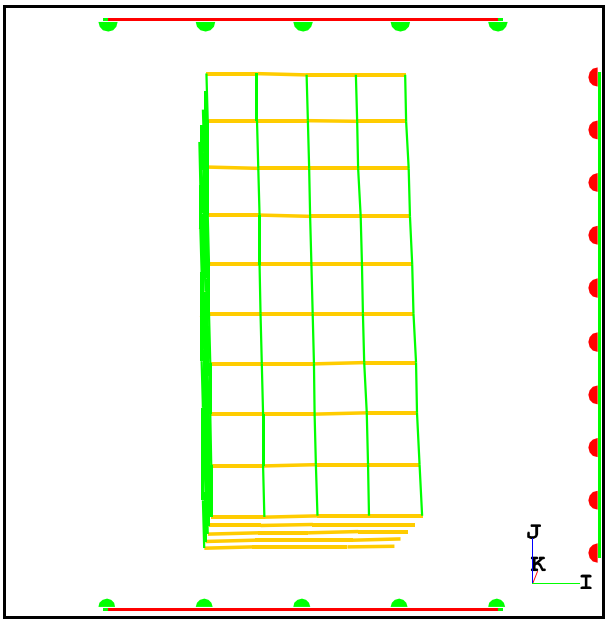


Figure 265 j-partitions

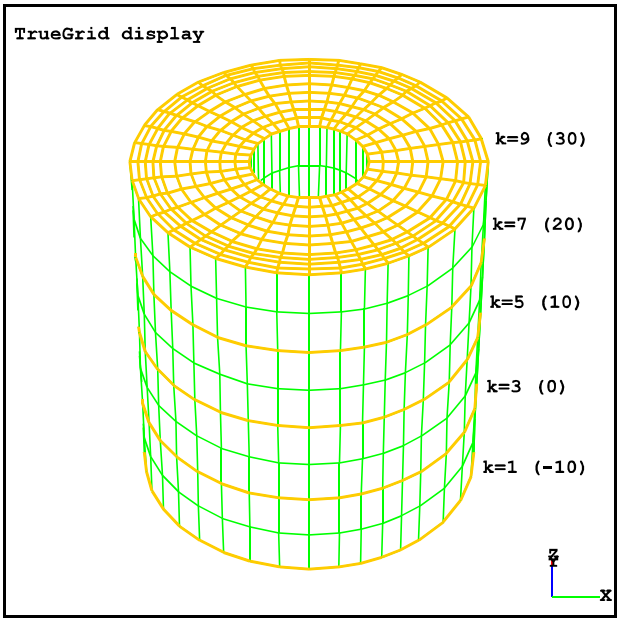


Figure 266 k-partitions and z-coordinates

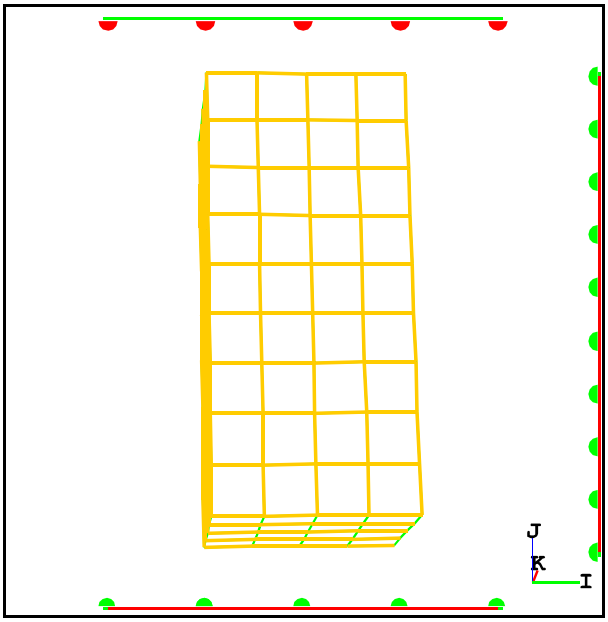


Figure 267 k-partitions

Remarks

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

When this command is issued, the present part is ended as if the **endpart** command was issued. Then this new part is immediately initialized.

Like **block**, this command transitions the code to the Part Phase. This is the standard way to generate a part. In order to complete the Part Phase and return to the Control Phase, use the **endpart** or **control** command. In order to complete the Part Phase and return to the Merge Phase, use the **merge** command. In order to abort the part and return to the Control Phase, use the **abort** command.

Six lists of numbers follow the **cylinder** command. The first three lists consist of integers, each list terminated with a semi-colon. The second three lists consist of real numbers, each list optionally terminated by a semi-colon. The first list of integers must start with a `1` or `-1`. The integers that follow must be zero or have absolute value greater than the absolute values of the integers that preceded it in that list. These are the number of nodes to create in the first dimension of the computational mesh. A positive integer indicates that there will be a partition at that nodal index in the first dimension of the computational mesh, which can be referenced by most commands in the Part Phase. These partitions are used to break the part into multiple structured blocks. Since most part commands reference only the reduced indices and not the nodal indices, it is a trivial matter to change the density of the mesh by only changing lists of nodal indices in the part definition. When positive nodal indices are used, solid elements are created. If you are generating solid elements exclusively, you may prefer the alternative method of specifying the number of elements in each list by first issuing the **partmode** command. A negative nodal index in the list also produces a partition in the mesh with a nodal index corresponding to the absolute value of the integer, but with shell elements created along that partition in the computational mesh. A zero in the list means there is a gap in the mesh, i.e. the part is not connected. The second and third lists have the same meaning applied to the second and third dimensions in the computational mesh, respectively.

The fourth list is a list of radial coordinates, one coordinate for each integer in the first list. This initializes each partition in the first dimension of the computation mesh. The fifth list consists of angular coordinates, corresponding in like manner to the second list of partitions in the second dimension of the computational mesh. The sixth list consists of z-coordinates, corresponding in like manner to the third list of partitions in the third dimension of the computational mesh.

The cylindrical coordinates relate to Cartesian coordinates by

$$x = r\cos(\theta) \qquad y = r\sin(\theta) \qquad z = z\ .$$

When the part is completed, **True*Grid*®** will apply this transformation to put the part in the global Cartesian coordinate system. By default, this means that the axis of the part's cylindrical coordinate system will be the same as the z-axis of the global coordinate system. You can select a local frame of reference by choosing the z-axis of this coordinate system to be any vector in the global coordinate

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

system by issuing the **cycorsy** command. Of course, after you finish making the part you can replicate or transform it into another part with whatever axis you like.

But for some uses of the **sf** and **sfi** commands, it is more convenient to define your part at its final location in the global coordinate system. If you define a part with projections and then transform it to its final location, the projection surfaces will have to be transformed from the final location to the place where the part is defined. If the **cylinder** command does not let you define your part at a convenient location, then you should use the **block** command. You can project a **block** part to make the same shape that a **cylinder** part would make.

Many computations that apply to the part will use cylindrical coordinates. An example of this is interpolations: an interpolation that will put points along a line for a **block** part will put points along an arc for a **cylinder** part if the angular coordinates at the end nodes are different.

Some part commands, such as **fd**, use physical coordinates in their arguments. For simplicity this manual names such command arguments in terms of Cartesian coordinates. For example, *x y z* would be arguments defining the physical coordinates of a point. However, the coordinate system of such arguments is always the same as the coordinate system of the relevant part. Thus if the part is set up by a **cylinder** command, you would give the physical coordinates of a point as its r-, θ-, and z- coordinates.

readmesh read a file containing a mesh

readmesh *format filename cmds* **endpart**

where

format can be:

nastran	is partially supported at this time
neutral	is partially supported at this time
dyna3d	is nearly fully supported
lsdyna	reads only nodes and elements
dynain	this is a file written by LS-DYNA
iges	reads a FEM model from an IGES file

filename is the path and file name of the formatted data

cmds can be

cvtab	to write the conversion tables from NASTRAN to **True*Grid*®**
exclude	exclude the NASTRAN model when writing the LS-DYNA
mate *mat*	used only for IGES where *mat* is the material number for all elements in the file.
mt *type mat*	used only for IGES. *Type mat* can be repeated any number of times.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

where *type* can be any or all of

beams
lshells for linear shells
qshells for quadratic shells
lbricks for linear bricks
qbricks for quadratic bricks
springs
dampers

where *mat* is the material number for all elements of type in the file. Use 0 to ignore that type of element from the IGES file.

ptmass *mass* used only for IGES

where *mass* is the mass at all point masses. Use 0 to ignore the point masses in the IGES file.

rigid *switch* used only for IGES

where *switch* can be

on to include rigid beams (default)
off to ignore rigid beams

cond *switch* used only for IGES

where *switch* can be

on to include nodal conditions (default)
off to ignore nodal conditions

ndcons *load_case_list* ; used only for IGES

where *load_case_list* is a list of the load cases to be combined

maplabel *list_maps* ; used only for IGES to ANSYS translations

where a single map is formed by a label, file name, and material Id

Remarks

This is a part, and it must be ended with the **endpart** command. This command will read in a formatted NASTRAN, PATRAN, DYNA3D, LS-DYNA, and IGES file. The data in any of these files will be added to the **True*Grid*®** internal data base. It can be written out in another format. Only some of the data is understood at this time. The list of features for each format is found below.

Material models, cross sectional properties for beams and shells, spring properties, and other specialized elements do not convert directly from one format to another. For this reason, you should redefine these features within **True*Grid*®**.

The loads, boundary conditions, etc. can be viewed using the **condition** command in the graphics menu while in the Merge Phase. They can be modified as well - see the command in the Merge Phase for that specific condition. Use the node, face, and element set features to select objects from

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

these parts and assign boundary conditions and properties while in the merge phase.

When a NASTRAN file is imported, **True*Grid*®** will report the number of features it translates into the **True*Grid*®** data base. That report may look like:

```
 NASTRAN objects included in the TRUEGRID data base:
      4153 node points
        32 beam elements
       333 linear shell elements
        21 quadratic shell elements
        11 linear brick elements - including tets and wedges
       357 quadratic brick elements - including tets and wedges
         2 beam element cross section definitions
         6 material models
         2 local coordinate systems defined
         4 load sets
                   45     324       3     655
           spring material properties
        22 springs
 There were       12 warning conditions from the NASTRAN deck
```

Load sets are similar to load curves in many of the dynamics codes and it will be necessary to define the appropriate load curves when porting to these codes. In the above example, if the output format is DYNA3D, then the load curves 45, 324, 3, and 655 should be defined.

The NASTRAN features which are supported in the READMESH command are:

CBAR	saved as a 2D element - cross sectional properties can be changed
CBEAM	saved as a 2D element - cross sectional properties can be changed
CBEND	saved as a 2D element - cross sectional properties can be changed
CELAS1	scalar and ground points are not supported, a PELAS must be referenced
CELAS2	scalar and ground points are not supported, this also produces a spring material definition
CHEXA	
CORD	
CPENTA	
CQUAD4	
CQUAD8	
CQUADR	
CROD	saved as a 2D element - cross sectional properties can be changed
CSHEAR	
CTETRA	
CTRIA3	

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

CTRIA6	
CTRIAR	
CTUBE	saved as a 2D element - cross sectional properties can be changed
FORCE	
FORCE1	
FORCE2	
GRDSET	
GRID	
MAT1	must be redefined if using another output format
MAT2	must be redefined if using another output format
MAT4	must be redefined if using another output format
MAT5	must be redefined if using another output format
MAT8	must be redefined if using another output format
MAT9	must be redefined if using another output format
MAT10	must be redefined if using another output format
MOMENT	
MOMENT1	
MOMENT2	
PBAR	must be redefined if using another output format
PBCOMP	must be redefined if using another output format
PBEAM	must be redefined if using another output format
PBEND	must be redefined if using another output format
PCOMP	must be redefined if using another output format
PELAS	must be redefined if using another output format
PLOAD	
PROD	must be redefined if using another output format
PSHEAR	must be redefined if using another output format
PSHELL	must be redefined if using another output format
PSOLID	must be redefined if using another output format
PTUBE	must be redefined if using another output format
RBE2	

A experimental feature with the **readmesh** command for NASTRAN files can be used in a limited way to import an existing NASTRAN model and add parts which can be merged to the nodes of the NASTRAN model (preserving the original node numbers) such that when the model is written out to an LS-DYNA keyword format, only the new nodes and elements are written out. This is the **exclude** option when reading a NASTRAN file. The **exclude** option means exclude objects from the **readmesh** when writing the output file. The is useful when building an add on to a model that already exists as a NASTRAN model that has been converted to LS-DYNA. If the **readmesh** command were able to read an LS-DYNA input and write it out to produce the exact model that was

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

raed in, then this exclude feature would not be useful.

No new nodes or elements will be given numbers that are already in use in the original NASTRAN model. This feature is to meet a specific need in a timely fashion. However, this feature may have a more general use if it is extended to all keyword input and output. This would make it possible to modify a model incrementally by creating add-on files which can be concatenated. It is not clear that such an improvement will be made. Comments on this feature will be appreciated.

There are two limitations to keep in mind. The **readmesh** command should be issued before any parts with nodes that should be merged to nodes from the NASTRAN input file. This will probably remain a limitation or feature. Secondly, only the new nodes and elements are protected from colliding with the objects in the NASTRAN input file. Materials, springs and dampers, sets, etc. must still be managed by the user.

The IGES option reads in FEM elements, properties, and loads from an IGES file. Currently, entity 134 (NODE), most element types for entity 136 (element type 1, 2, 3, 5, 6, 12, 13, 14, 15, 17, 18, 27, 29, 31, 32, 33, and 35), 406 (tabular data, form 11, property type 12 nodal loads/constraint, and 418 (nodal load/constraint) are supported. The material number may be set for all elements in the IGES file or individually for beams, linear shells, quadratic shells, linear bricks, quadratic bricks, springs and dampers. (Setting the material for an element type to zero causes TrueGrid to skip all elements of that type in the file.) Also, the mass for the point mass (entity 136 element type 31) may be set (a point mass of zero means skip). Rigid beams (entity 136 element type 32) and nodal loads/constraints (entity 418) may be skipped using the **rigid off** and **cond off** options, respectively. Nodal load cases may be selected using the **ndcons** option followed by the list of nodal load cases to be combined. Note that **cond off** will cause all nodal conditions to be ignored even if **ndcons** is set.

The **maplabel** option is used in conjunction with an ANSYS output to reference beam cross sections data from a file that the ANSYS code references. If the entity 406, form 11, ptype 5008 is found in the IGES file, then the referenced beam elements will be given a beam type of 188 and the appropriate file for the cross section data will be referenced. The **maplabel** option can be repeated any number of times.

Related to the **dynain** option in the **readmesh** command is the **dynain** output format. One will usually read a **dynain** file using the **readmesh** command, modify the data using the **smags** and **delem** commands, and write the results back to a file using the **dynain** and **write** commands. Node and element numbers are preserved.

blude extrude a set of polygons

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

blude *direction face_set_name*
i_indices ; j_indices ; k_indices ; x_coordinates y_coordinates z_coordinates
where

direction	is the face of the block, where the extrusion begins and can be one of:
1	for the minimum i-face
2	for the maximum i-face
3	for the minimum j-face
4	for the maximum j-face
5	for the minimum k-face
6	for the maximum k-face
face_set_name	is the name of the face set to be extruded
i_indices, *j_indices*, and *k_indices*	are lists of indices (not reduced indices). Each list of indices takes the form i1 i2 ... in, where i1=1 and |i1| < |i2| < ... < |in|. Normally each of these indices is a positive integer. But a negative integer can be used to create a shell surface, and a zero to separate parts of the mesh.
x,y,z_coordinates	are lists of values of physical coordinates, one for each index value: an x coordinate for each i index, a y coordinate for each j index, and a z coordinate for each k index. These define the physical locations of every region interface.

Remarks

This command extrudes a set of polygons. It is based upon the **block** command and an understanding of that command is essential to understanding this one. This extrusion is done by the construction of a block part with no deleted regions. The polygons in the face set are extruded, one polygon at a time, by following the mesh lines formed by the block part. See also the **block** command. The set of polygons is defined using a face set. See also the **fset** command. The block part is discarded. Its only purpose is to define the flow lines for the extrusion of the polygons. Be sure, when you are building the block part, that one of its 6 outer faces completely covers the face set of polygons.

This command transitions the code to the Part Phase. In order to complete the Part Phase and return to the Control Phase, use the **endpart** or **control** command. In order to complete the Part Phase and return to the Merge Phase, use the **merge** command. In order to abort the part and return to the Control Phase, use the **abort** command.

This part is difficult to use. Its application is to a small class of problems and many conditions must

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

be near perfect for it to work properly. There is no guarantee that it will work for a problem. This part is not recommended for the casual user.

Example

In this example, a simple NASTRAN ship model is read using the **readmesh** command. This mesh is unstructured because there is a small region of the hull with triangles. A face set is constructed using the mouse to select the faces on the starboard side. This face set is then used to construct a face surface for projection and for the **blude** command. An intra-part **bb** command with a normal offset is used in the construction of the liner to form an orthogonal part. This liner is needed because a wedge part just below the keel is needed to round off the keel. Only then can a good fluid mesh be extruded to the outer cylinder wall of the region of interest. It would be possible to use the intra-part **bb** command with an offset to form the boundary layer, however in this example, for graphical purposes, the boundary region was made too large for the normal offset feature in the **bb** command to work effectively. The normal offset feature is only good for short distances relative to the changes in curvature on the master surface. This entire model is too complicated to have the input files included in this text. The input files are available.

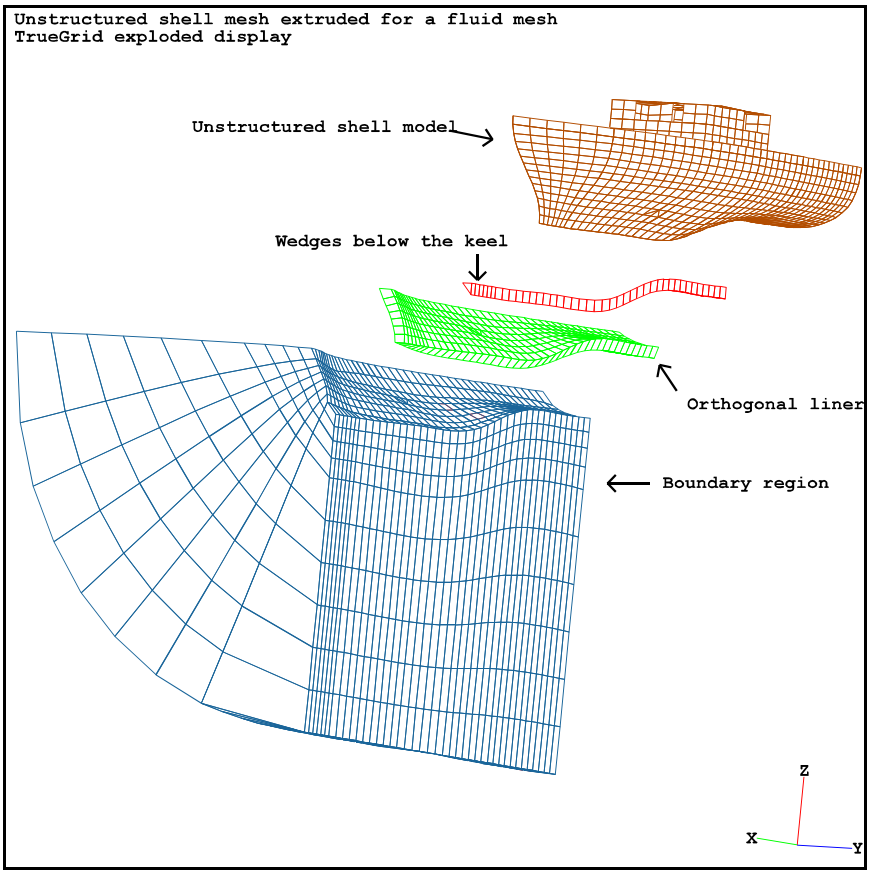


Figure 268 Shell structure extruded forming a solid

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

meshscal scale up the mesh density for all parts

meshscal *scale*

where *scale* is a positive integer

Remarks

This will increase the mesh density uniformly in all 3 directions by an integer scale factor. Typically a factor of 2 (8 times as many bricks, 4 times as many shells, 2 times as many embedded beams are the result) or 3 is sufficient. This should come first before any parts are made. This cannot be used if the **update** command is used. The default is 1.

beam initialize a beam part in Cartesian coordinates

beam *options* **endpart**

where

options are any of the following (this may be continued across many lines)

abort

bm $node_1$ $node_2$ *number_of_elements material cross-section* $node_3$ *;*

to create a sequence of beam elements, where:

$node_1$ and $node_2$	form the end nodes of the sequence of beams,
number_of_elements	is the number of beam elements,
material	is the material number,
cross-section	is a cross-section number, and
$node_3$	is the local coordinate system orientation node.

cy *radius angle z-coordinate constraints ;* for cylindrical coordinates

where:

dx	for x-displacement,
dy	for y-displacement,
dz	for z-displacement,
rx	for rotation about the x-axis,
ry	for rotation about the y-axis,
rz	for rotation about the z-axis,

followed by a value of

0	for initialization to no constraint
1	for constraint

jt *joint_# joint_node_# beam_node_# joint_increment ;* for joint nodes

joint_# for the **True*Grid*®** joint definition number,

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

joint_node_# each joint type consists of a certain number of nodes, which are numbered in a specified way for **True*Grid*®** (consult the user's manual). You must specify which node of the joint this beam node must become,

beam_node_# this is the **True*Grid*®** beam node number for the current beam part,

joint_increment this is used when there are replications and indicates how the joint_no should be adjusted for replications

lct *#_transforms ; first_transforms ; ... ; last_transforms ;* transformations

lrep *list_local_transformation_# ;* for replication of the beams

rt *x-coordinate y-coordinate z-coordinate constraints ;* for Cartesian coord's

where the constraint list is composed of

dx for x-displacement
dy for y-displacement
dz for z-displacement
rx for rotation about the x-axis
ry for rotation about the y-axis
rz for rotation about the z-axis

followed by a value of

0 for initialize to no constraint
1 for constrain

si *interface_# option_list ;* for sliding interface nodes

where an option can be

ffn normal_force
ffs shear_force
enf e_normal_force
esf e_shear_force

sp *radius angle_1 angle_2 constraints ;* for spherical coordinates

where the constraint list is composed of:

dx for x-displacement
dy for y-displacement
dz for z-displacement
rx for rotation about the x-axis
ry for rotation about the y-axis
rz for rotation about the z-axis

followed by a value of

0 for initialize to no constraint
1 for constrain

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

This command may be issued in any phase. If it is issued in the block or cylinder Part Phase, this command will end that part and create this new part. The implementation of the beam part is not interactive, although it is fully functional. It is provided to be nearly compatible with the INGRID beam part. Please note the differences in the two parts.

The intent here is to first create a set of nodes, one node at a time using either **rt**, **cy**, or **sp**. Each node is assigned a set of nodal constraints as part of the definition of the node. No constraints are needed. Usually beam nodes are merged to other nodes in the model, so that they will then inherit the constraints of the other nodes that they are merged to. These nodes are then tied together to form beam elements using the **bm** command. The nodes are referenced by their sequence number within the beam command. Some simulation codes, such as DYNA3D, require a third node in a beam element definition to define the orientation of the beam. Use the **bsd** command to define the cross sectional properties of the beam. Use the appropriate material definition command to assign properties to the numbered material referenced in the **bm** command.

Coordinates are referenced in their respective systems but all interpolation is done in Cartesian coordinates.

See the **jd** and **jt** command within the block part to understand the beam **jt** command. Joints require at least two nodes. The **jd** command is used to define joint properties. The nodes of a joint are numbered in a set way. You can flag nodes created in a beam part for use in a joint using the **jt** command

cbeam initialize a beam part in cylindrical coordinates

cbeam *options* **endpart**

where

options are the same as in **beam** except that the interpolation of node points in the **bm** option is done in a cylindrical coordinate system.

Remarks

See the **beam** command.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

ap add a part to the picture

ap *part_number*

Remarks

The display list determines what objects are in the picture. A part in the display list is referred to as being active. Part number 0 refers to the present part. If the selection is the present part, then it can be viewed in full. If it is any other part, it can only be viewed as a wire frame.

aps add a list of parts to the picture

aps *part_list ;*

Remarks

The display list determines what objects are in the picture. A part in the display list is referred to as being active. Part number 0 refers to the present part. If one of the active parts is the present part, then it can be viewed in full. All other parts are only viewed as a wire frame. A list of part numbers can include a sequence of part numbers with only the first and last numbers separated by a colon (e.g. aps 2:5;).

dap display all parts

dap *(no arguments)*

Remarks

The display list determines what objects are in the picture. A part in the display list is referred to as being active. The present part will entirely be shown. The previously generated parts will be wire frame only.

dp display one part in the picture

dp *part_number*

This command will display one, and only one, part.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

dps display a set of parts in the picture

dps *part_list* ;
where
part_list is a blank-delimited list of part numbers

Remarks

The display list determines what objects are in the picture. Part number 0 refers to the present part. A part in the display list is referred to as being active. If a selection is the present part, then it can be viewed in full. If it is any other part, it can only be viewed as a wire frame. A list of part numbers can include a sequence of part numbers with only the first and last numbers separated by a colon.

pinfo part information

pinfo *(no arguments)*

rap remove all parts from the picture

rap *(no arguments)*

Remarks

The display list determines what objects are in the picture. A part in the display list is referred to as being active.

rp remove one part from the picture

rp *part_number*
where
part_number is a part number.

Remarks

The display list determines what objects are in the picture. A part in the display list is referred to as being active. Part number 0 means the present part.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

rps remove a set of parts from the picture

rps *part_list ;*

Remarks

The display list determines what objects are in the picture. A part in the display list is referred to as being active. Part number 0 refers to the present part. A list of part numbers can include a sequence of part numbers with only the first and last numbers separated by a colon.

3. Motion

These commands apply initial velocities to all parts. The **rotation** or **velocity** commands in the Part Phase can override these commands for just one part. The **ve** or **vei** commands in the Part Phase can override these commands for selected regions of the part.

rotation global initial velocities as a rigid body rotation

rotation x_0 y_0 z_0 *x_rotation y_rotation z_rotation*

where

(x_0,y_0,z_0)	is any point on the axis of rotation, and
(x_rotation,y_rotation,z_rotation)	is the rotation vector in radians per unit time.

Remarks

By default, an initial body rotation is assigned to all parts defined after this command. Use the units expected by the simulation code, as **True*Grid*®** does no conversions. (**True*Grid*®** is dimension less.)

Examples

The following example assigns initial velocities which rotate the shell cylinder mesh about its axis of symmetry. The velocity magnitude is 0.2 because the radius of the cylinder is 2.

```
sd 1 cy 0 2 2 1 0 0 2
rotation 0 2 2 .1 0 0
block 1 11;-1 -11;-1 -11;-1 1 1 3 1 3
sfi ; -1 -2; -1 -2;sd 1
endpart
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

velocity global initial velocities as a rigid body translation

velocity *x_velocity y_velocity z_velocity*
where
(x_velocity,y_velocity,z_velocity) is a velocity vector.

Remarks

An initial rigid body velocity is assigned to all parts defined after this command. Use the units expected by the target simulation code as **True*Grid*®** does no conversions.

4. Boundary Conditions and Constraints

The **detp** command defines detonation points or lines. The **jd** command defines joint. The **jtinfo** command writes information about joints. The **lsys** command defines a local coordinate system for the **lb** command. The **lsysinfo** command lists all of the local coordinate systems. The **plane** command defines a boundary plane. The **plinfo** command writes information about defined boundary planes. The multiple point constraint (**mpc**) and tracer point (**trp**) commands are discussed in the part and assembly sections of this manual because they function differently in each phase of the model generation.

detp detonation points or lines

detp *material options ;*
where
material is the material number and
options is a list of any of the following:
time *detonation_time*
point *x y z*
lnpt $x_1\ y_1\ z_1\ x_2\ y_2\ z_2$ *#_detonators*

Remarks

This creates detonation points and lighting times for high explosives. Use the **point** option if there is only one detonation point. Otherwise, use the **lnpt** option for a line of detonation points equally spaced.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

jd joint definition

jd *joint_# type options ;*

where *joint_#* must be between 1 and 300

where *type* can be (default is shared constraints)

sj	for spherical joint
rj	for revolute joint
cj	for cylindrical joint
pj	for planar joint
uj	for universal joint
tj	for translational joint
sw	to constrain nodes as a spotweld

where *option* can be

pnlt *joint_penalty*	for joint penalty
repe *#_joint_replications*	for joint replications
dx	to specify shared x-displacement between the nodes
dy	to specify shared y-displacement between the nodes
dz	to specify shared z-displacement between the nodes
rx	to specify shared x-axis rotation between the nodes
ry	to specify shared y-axis rotation between the nodes
rz	to specify shared z-axis rotation between the nodes

Remarks

Use this command first to define the properties of a numbered joint. Then use the **jt** command to identify the nodes used in the joint.

There are two types of joints. **Sj** , **rj**, **cj**, **pj**, **uj**, and **tj** form the first type and "nodes constrained together" form the second type. See the diagrams below for the relationship between the numbered nodes of the first type.

The **pnlt** option is used for the **DYNA3D**, **LSDYNA** and **NIKE3D** output only in the first type of joint definition. The **repe** option makes it possible to create a sequence of duplicate joint definitions starting with the one specified by the **jd** command.

Up to 16 nodes may be assigned to a joint. If the joint is of a type that requires fewer nodes, **True*Grid*®** will ignore any superfluous nodes assigned to that joint. If there are more than 16 nodes to share constraints, use the **mpc** command.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

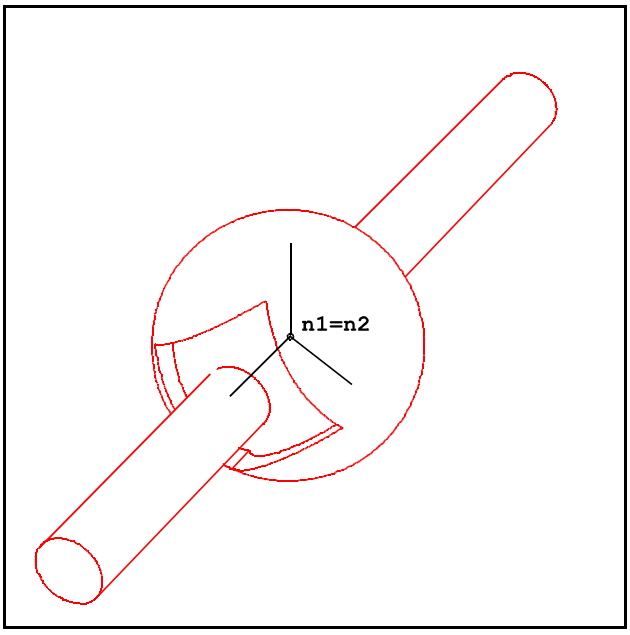


Figure 269 Spherical Joint

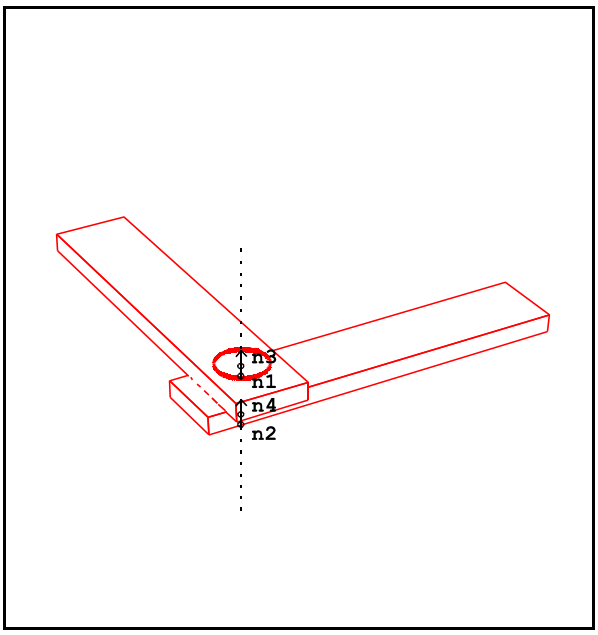


Figure 270 Revolute Joint

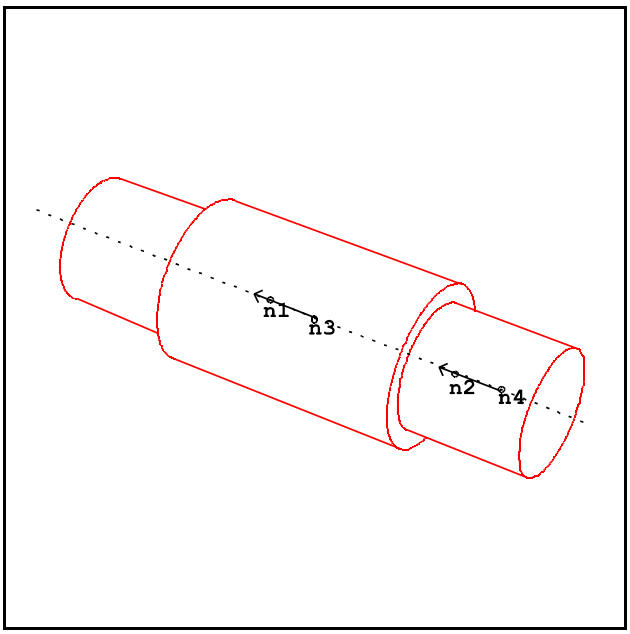


Figure 271 Cylindrical Joint

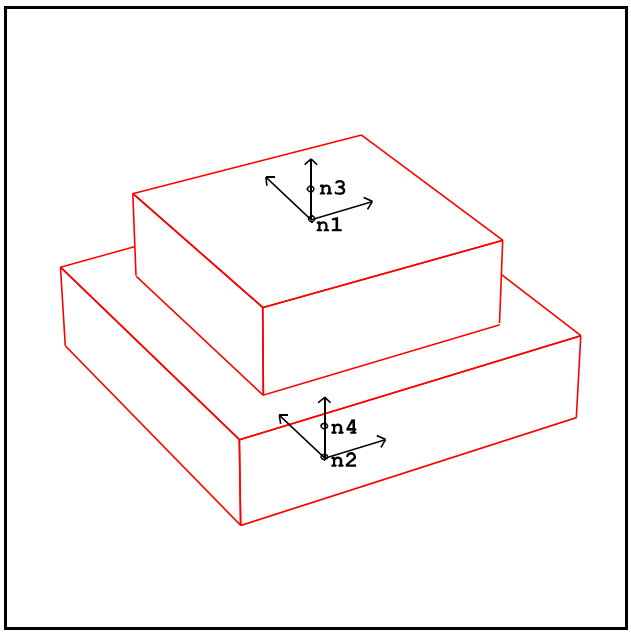


Figure 272 Planar Joint

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

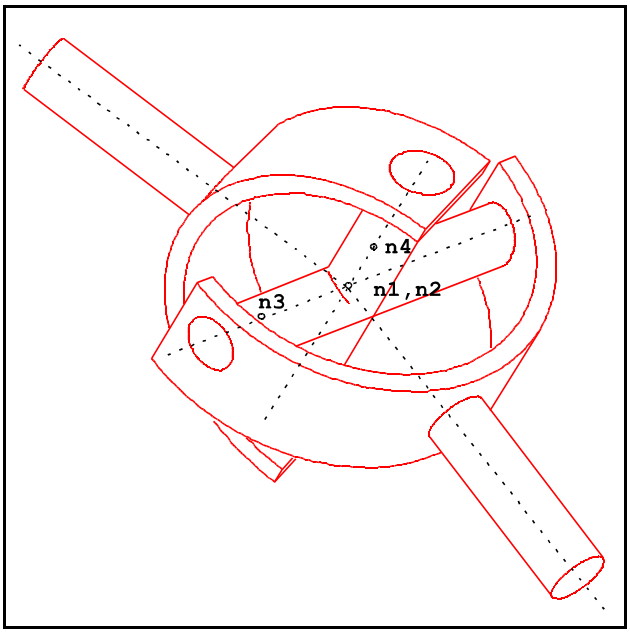


Figure 273 Universal Joint

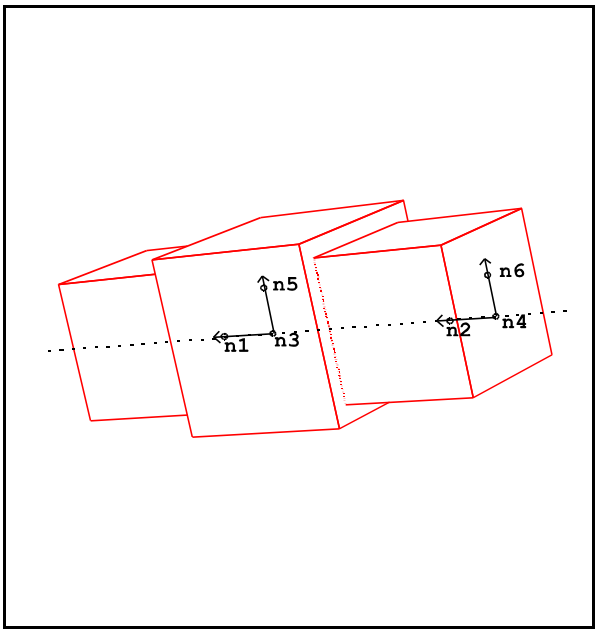


Figure 274 Translational Joint

jtinfo write information about joints

jtinfo *<no arguments>*

Remarks

This command lists all of the joint definitions in the model.

lsys define a local coordinate system for the lb command

lsys *system transformations ;*

where

system is the "system number" which **lb** will use to identify this local coordinate system, and

transformations is a list of transformations that define the local coordinate system, chosen from the following:

mx *x_offset*	for translation in x
my *y_offset*	for translation in y
mz *z_offset*	for translation in z

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

v *x_offset y_offset z_offset* for translation in an arbitrary direction
rx θ for rotation about the x axis
ry θ for rotation about the y axis
rz θ for rotation about the z axis
tf *origin x-axis y-axis*
where each of the arguments consist of a coordinate type followed by coordinate information:
rt *x y z* for Cartesian coordinates
cy *rho theta z* for cylindrical coordinates
sp *rho theta phi* for spherical coordinates
pt *c.i* for a label of a labeled point from a 3D curve
pt *s.i.j* for a label of a labeled point from a surface
ftf *1st_origin 1st_x-axis 1st_y-axis 2nd_origin 2nd_x-axis 2nd_y-axis*
where each of the arguments consist of a coordinate type followed by coordinate information:
rt *x y z* for Cartesian coordinates
cy *rho theta z* for cylindrical coordinates
sp *rho theta phi* for spherical coordinates
pt *c.*i for a label of a labeled point from a 3D curve
pt *s.i.j* for a label of a labeled point from a surface
inv to invert the present transformation

Remarks

This command is used before using the **lb** command. This command defines the local coordinate system to constrain a set of nodes in a coordinate system different from the global coordinate system.

lsysinfo list all of the local coordinate systems

lsysinfo *<no arguments>*

Remarks

This command lists all of the local coordinate systems defined using the **lsys** command.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

plane define a boundary plane

plane *plane_#* x_0 y_0 z_0 x_n y_n z_n *tolerance type*

where *type* can be

symm	for symmetry plane
syf	for symmetry plane with failure
ston *features*	for stonewall with optional features

where a *feature* can be

stick *stick_condition*
limit x_c y_c z_c x_1 y_1
move *mass initial_velocity*
penalty *penalty_stiffness*
dlcv *velocity_ld_curve_#*

Remarks

The point (x_0, y_0, z_0) is on the plane with a normal vector (x_n, y_n, z_n).

This command is used to define nodal constraints for nodes on a symmetry plane. Some simulation codes also have a symmetry with failure and a stone wall feature which is supported by this command.

Nodes are automatically selected for the symmetry plane constraint if they are within the specified tolerance of the plane. The tolerance is not used for the symmetry plane with failure and the stone wall. Use the **syf** and **syfi** commands to select nodes on the symmetry plane with failure. Use the **sw** and **swi** commands to select nodes on the stone wall. These planes are numbered so that there can be multiple stone wall and symmetry plane with failure conditions.

The symmetry feature is complicated, depending on the type of plane and the simulation code. If the symmetry plane is parallel to one of the planes where x=0, y=0, or z=0, then the nodes on the symmetry plane are assigned constraints in the global coordinate system. These types of symmetry planes are referred to as canonical symmetry planes and are equivalent to the following constraints:

plane parallel to x=0: x-displacement, y-rotation, z-rotation
plane parallel to y=0: y-displacement, x-rotation, z-rotation
plane parallel to z=0: z-displacement, x-rotation, y-rotation

Some simulation codes support symmetry planes other than these canonical forms, such as a symmetry plane where x=y (45 degrees). In this case, a different option is used when the model is output. A complication occurs when a set on nodes fall on more than one of these types of symmetry

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

planes.

For DYNA3D, when a node is found on 2 non-canonical distinct symmetry planes, then a sliding boundary plane of vector type is automatically created for this node. If a node is found on 3 or more distinct symmetry planes, it has all of its DOFs constrained. If a node was assigned nodal boundary constraints orthogonal to the sliding boundary plane to which it is assigned, those nodal boundary constraints will be ignored and a warning message will be written.

For LS-DYNA, the canonical (x,y or z) planes are handled as global constrains in the *NODES section. The non-canonical symmetry planes are handled with the *BOUNDARY_SPC_SETS and *BOUNDARY_SPC_NODE keywords along with the *DEFINE_COORDINATE_SYSTEM.

For ABAQUS, ANSYS, NASTRAN and NE/NASTRAN, nodes on non-canonical symmetry planes are constrainted in local coordinate systems.

plinfo write information about defined boundary planes

plinfo <*no arguments*>

Remarks

This command lists all of the planes defined in the **plane** command.

5. Radiation and Temperatures

temp global default constant temperature

temp *temperature*

Remarks

A temperature is assigned to all parts defined after this command. Use the units expected by the target simulation code as **True*Grid*®** does no conversion.

For the NASTRAN and NE/NASTRAN output, the last value is used for the TEMPD command.

For DYNA3D output, this produces the temperatures for the temperature option 2.

For LS-DYNA output, this produces the *LOAD_THERMAL_CONSTANT_NODE.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Use the **te** or **tei** command to change the default constant temperature for a specific region of the mesh.

bfd bulk fluid definition

bfd *id_# material_# volume*

where

id_#	identification number of this definition to be used by the **bf** command
material_#	material number associated with the bulk fluid
volume	total volume of the material

Remarks

Use **bfd** to define the material number and total volume of a bulk node.

6. Springs, Dampers, and Point Masses

There are two ways to generate springs or dampers. The **spdp** command is used to create an array of springs or dampers between two disconnected faces of the mesh, one spring between each pair of corresponding nodes. The **spring** command is used to create a single spring between two nodes. The **spdp** command is only found in the part phase. The **spring** command is found in both the part and merge phase, but the syntax differs. Both of these commands refer to a spring definition number. The **spd** command is used to define the properties of a spring and assign a definition number to these properties.

There are two ways to generate a point mass. The **npm** command creates a new node with a point mass. Since this node is not connected to the rest of the model, you must take additional steps to make sure it is connected through merging or building a beam or spring using this node. The **pm** command can also be used to assign a mass to an existing node. Although the **npm** command may appear to be a global command, it causes point mass replications when found in the part phase when the part is replicated, distinguishing it from the **npm** command found in the merge phase. The **pm** command is found in both the part and merge phase, but their syntax differs.

Springs, dampers, and point masses are replicated when they are specified within a part and the part is replicated. When the spring definition number is incremented in the spring command, make sure that you use the **spd** command to define each spring property.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

spd define the properties of a set of springs or dampers

spd *spring/damper_# option type parameters*

where

option can be

dro *flag*

where flag can be:

0 for a linear spring/damper

1 for a torsional spring/damper

dmf *dynamic_factor*

tv *test_velocity*

cl *clearance*

fd *failure_deflection*

dlc *limit_compression*

dlt *limit_tension*

where

type is the spring or damper's material model and

parameters is a list of corresponding parameters, as in the following:

le *stiffness* for linear elastic

lv *damping* for linear viscous

iep *elastic tangent yield* for isotropic elastic

ne *ld_curve_#* for nonlinear elastic

nv *ld_curve_#* for nonlinear viscous

gn *loading_# unloading_# hardening tension compression* for general nonlinear

dhpt for a dashpot

mv for a three parameter maxwell viscoelastic

itc for a inelastic tension or compression only

se *elastic_value damping stress* for scalar elastic

mus *l0 vmax sv a fmax tl tv fpe lmax ksh* for muscle

where

l0	for initial muscle length
vmax	for maximum CE shortening velocity
sv	for scale factor for Vmax vs. activs state
a	for activation level vs. time function
fmax	for peak isometric force
tl	for active tension vs. length function
tv	for active tension vs. velocity function
fpe	for force vs. length function
lmax	for relative length

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

ksh for exponential rise constant

Remarks

A spring or damper is defined using either the **spdp** command forming a set of springs/dampers between two surfaces, or using the **spring** command to create a single spring at a time. In each case, the definition of a spring includes a reference to a material definition **spd** number.

ANSYS, use linear elastic, linear viscous (damper), isotropic elastoplastic, or the dashpot.

DYNA3D, use linear elastic, linear viscous (damper), isotropic elastoplastic, nonlinear elastic, nonlinear viscous, general tabulated nonlinear, and dashpot.

LS-DYNA, use linear elastic, linear viscous (damper), isotropic elastoplastic, nonlinear elastic, nonlinear viscous, general tabulated nonlinear, dashpot, three parameter Maxwell viscoelastic, inelastic tension or compression only, and muscle. LS-DYNA3D options also include **dro**, **dmf**, **tv**, **cl**, **fd**, **dlc**, and **dlt**.

LS-NIKE3D, use linear elastic, linear viscous (damper), isotropic elastoplastic, nonlinear elastic, and nonlinear viscous.

MARC, use linear elastic, linear viscous (damper), isotropic elastoplastic, or the dashpot.

NASTRAN, use scalar elastic.

NE/NASTRAN, use scalar elastic.

NIKE3D, use linear elastic, linear viscous (damper), isotropic elastoplastic, nonlinear elastic, nonlinear viscous, general tabulated nonlinear, and dashpot.

If the output option has been selected prior to using the dialogue box to make a selection, only the options available to that output option will be displayed in the dialogue box.

Example

```
nastran
spd 1 se 1 .1 2.1
block 1 6;1 6;1 6;-1 1 -1 1 -1 1
spdp 1 1 2 2 2 2 1 m 1 ;
block 1 6; 1 6; 1 6;-1 1 -1 1 1.1 3.1
spdp 1 1 1 2 2 1 1 s ; ;
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
merge
write
```

spinfo write information about springs and dampers

spinfo (no arguments)

7. Interfaces

bbinfo block boundary interface information

bbinfo

Remarks

Only the master side of the interface appears in this table. Additional information is printed when first entering the merge phase and performing nodal merging. Master interfaces defined within a part are not included in the table while in that part generation phase. Only after that part is completed will its master block boundary interfaces appear in the table.

Example

The following commands form 7 parts. The first part is a simple cylinder formed with a single block. Each of the faces of this part forms master side of another block boundary. Each of the subsequent parts are glued to this center part using the block boundary interface command.

```
sd 1 sp 0 0 0 1                          c sphere surface
block 1 11; 1 11; 1 11;-1 1 -1 1 -1 1 c part 1
sfi -1 -2; -1 -2; -1 -2;sd 1             c project to sphere
bb 1 1 1 1 2 2 1;                        c master BB 1
bb 2 1 1 2 2 2 2;                        c master BB 2
bb 1 1 1 2 1 2 3;                        c master BB 3
bb 1 2 1 2 2 2 4;                        c master BB 4
bb 1 1 1 2 2 1 5;                        c master BB 5
bb 1 1 2 2 2 2 6;                        c master BB 6
block 1 11;1 11;1 11;-3 -1 -3 3 -3 3  c part 2
bb 2 1 1 2 2 2 1;                        c slave BB 1
block 1 11;1 11;1 11;1 3 -3 3 -3 3    c part 3
bb 1 1 1 1 2 2 2;                        c slave BB 2
block 1 11;1 11;1 11;-3 3 -3 -1 -3 3  c part 4
bb 1 2 1 2 2 2 3;                        c slave BB 3
block 1 11;1 11;1 11;-3 3 1 3 -3 3    c part 5
bb 1 1 1 2 1 2 4;                        c slave BB 4
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
block 1 11;1 11;1 11;-3 3 -3 3 -3 -1  c part 6
bb 1 1 2 2 2 2 5;                     c slave BB 5
block 1 11;1 11;1 11;-3 3 -3 3 1 3    c part 7
bb 1 1 1 2 2 1 6;                     c slave BB 6
merge
```

The **bbinfo** command then produced this table of master block boundaries.

```
 Master Block Boundary Interface Table
 BB number      1 is     11 by     11 nodes
 BB number      2 is     11 by     11 nodes
 BB number      3 is     11 by     11 nodes
 BB number      4 is     11 by     11 nodes
 BB number      5 is     11 by     11 nodes
 BB number      6 is     11 by     11 nodes
```

getbb retrieve a block boundary from a part file

getbb *filename source_BB_# TG_alias_BB_#*

where

filename	name of the part file
source_BB_#	block boundary identification number in the part file
TG_alias_BB_#	new identification number for this block boundary

Remarks

Use the **savepart** command to save the part data including the master block boundary information to a file. The **getbb** command makes it possible to retrieve the block boundary information from this file without having to re-generate the entire part used to create the master side of the block boundary interface.

This feature can be used to define the connections between portions of a model so that a large model can be built by more than one individual. A shell part can be used to form such an interface by forming a cross section of the model. This splits the model. Then each individual must match this cross section as they build their portion of the mesh. Care must be taken to assure that the interface can be matched by both sides of the model.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

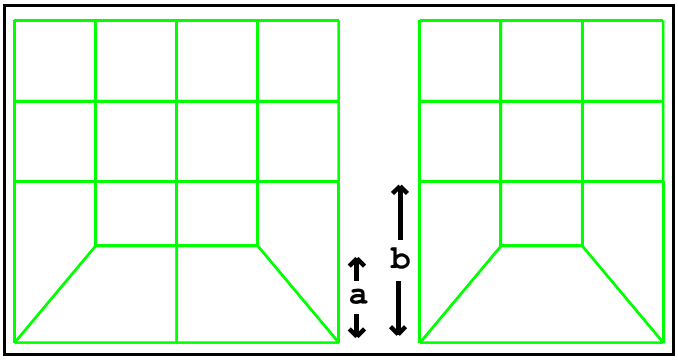
inttr block boundary transition element interpolation factor

inttr *α*

Remarks

The interior elements of a transition block are interpolated using a parameter between 0 and 1. The default for this parameter is 0.5. Referring to the diagram,

$$\alpha = a / b$$



Transition parameter

Example

```
inttr .6
block 1 5 0 6 9;1 3;-1;1 5 0 6 9 1 3 0 c part 1
bb 1 1 1 2 1 1 1;bb 4 1 1 5 1 1 2; c master BBs
block 1 3 0 4 5;1 2;-1;1 5 0 6 9 -1 1 0 c part 2
trbb 1 2 1 2 2 1 1;trbb 4 2 1 5 2 1 2; c slave transitions
merge
```

mbb master block boundary from point data

mbb *n #_rows #_columns x1 y1 z1 ... xn yn zn trans ;*

where

n	is the block boundary interface number
#_rows	is the number of rows in the table of coordinates
#_columns	is the number of columns in the table of coordinates
xi yi zi	are the coordinate triples for each node point
trans	is a coordinate transformation

Remarks

This creates a master block boundary from a table of coordinates. This feature is useful when a mesh is imported with the **readmesh** command. A master block boundary can be extracted from this part by picking nodes and using the ***F7*** function key to print out the coordinates. Care must be taken to select a block region and to pick the nodes in the correct order.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
mbb 10 3 3
 1 3  0  2  3.1 .25 3   3 0
.9 2 .25 2  2  -.25 3.1 2 .25
 1 1  0  2 .9   .25 3   1 0 ;
```

sid sliding interface definition

sid *slide_# option_list* ;

where *option_list* consists of

- **tied** for tied sliding surface (DYNA3D,NIKE3D)
- **sl** for sliding only (DYNA3D,NIKE3D)
- **sv** for sliding with voids (DYNA3D,NIKE3D)
- **single** for single sided slide surface (DYNA3D,NIKE3D)
- **dni** for discrete nodes impacting surface (DYNA3D)
- **dnt** for discrete nodes tied to surface (DYNA3D)
- **sets** for shell element edge tied to shell element surface (DYNA3D)
- **nsw** for nodes spot welded (DYNA3D)
- **break** for tie-break interface (DYNA3D)
- **owsv** for one way sliding with voids (DYNA3D)
- **dummy** is only used to insure that nodes in this interface will not be merged
- **sand** *options* for the Slide Surface with Adaptive New Definitions (DYNA3D)

  where *options* can be

  sms *slave_material_list* ;

  mms *master_material_list* ;
- **auto** for automatic contact (DYNA3D)
- **inter** for Interface elements (ABAQUS)
- **tcrs** *thermal_contact_resistance* for thermal sliding with thermal contact resistance (TOPAZ3D)
- **rebar** *options* to define properties of REBAR 1D sliding interface (DYNA3D)

  where *options* can be any of the following:

  rbrad radius

  rbstr strength

  rbshr modulus

  rbumax displacement

  rbexp exponent

  rbibond non-negative_number
- **pnlt** *penalty_factor* for sliding penalty (NIKE3D)

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

fric *friction_factor* for static coefficient of friction (DYNA3D,NIKE3D,ABAQUS)

kfric *kinetic_coefficient_of_friction* for kinetic coefficient of friction (DYNA3D,NIKE3D)

decay *exponential_decay_coefficient* for exponential decay coefficient (DYNA3D,NIKE3D)

pen for small penetration flag (DYNA3D)

sfif for slave to be printed in force file (DYNA3D)

mfif for master to be printed in force file (DYNA3D)

pnlts *slave_penalty_factor* for slave penalty factor (DYNA3D)

pnltm *master_penalty_factor* for master penalty factor (DYNA3D)

bwmrad *#_facets* to set the bandwidth minimization radius (NIKE3D)

fric2 *friction_factor* for the anisotropic friction coefficient (ABAQUS)

stif *stiffness* for stiffness in stick (ABAQUS)

essl *stress* for the equivalent shear stress limit (ABAQUS)

penmax *distance* to set the small penetration search distance

iaug *flag* to set the augmentation flag

where the *flag* can be

1	to augment until convergence tolerance is satisfied
0	for no augmentations (penalty method)
-n	for the number of augmentations per step

altoln *tolerance* to set the normal direction convergence tolerance

altolt *tolerance* to set the tangential direction convergence tolerance

tkmult *multiplier* to set the tangent stiffness multiplier

dtime *time* to set the interface death time

bury *time* to set the interface burial time

concon *conductance* contact conductance

radcon *conductance* radiation conductance

lsdsi *type options* ; for LS-DYNA sliding interfaces (LS-DYNA)

where *type* can be:

1	for Sliding without penalties
p1	for Symmetric sliding with penalties
2	for Tied
3	for Sliding, impact, friction
a3	for Sliding, impact, friction, no segmentation orientation
4	for Single surface contact
5	for Discrete nodes impacting surface
a5	for Discrete nodes impacting surface, no segmentation orientation
6	for Discrete nodes tied to surface

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

7 for Shell edge tied to shell surface
8 for Nodes spot welded to surface
9 for Tiebreak interface
10 for One way treatment of sliding, impact, friction
a10 for One way treatment, no segmentation orientation
13 for Automatic single surface with beams and arbitrary orientations
a13 for Automatic single surface with beams and arbitrary orientations with extra search for airbag contact
14 for Surface to surface eroding contact
15 for Single surface eroding contact
16 for Node to surface eroding contact
17 for Surface to surface symmetric/asymmetric constraint method
18 for Node to surface constraint method (Taylor and Flanagan 1989)
19 for Rigid body to rigid body contact with arbitrary force/deflection curve
20 for Rigid nodes to rigid body contact with arbitrary force/deflection curve
21 for Rigid body to rigid body contact with arbitrary force/deflection curve (one way treatment)
22 for Single edge treatment for shell surface edge to edge treatment
23 for Simulated draw bead
24 for Automatic surface to surface tiebreak
25 for Automatic one way surface to surface tiebreak
34 for Automatic general
35 for Automatic general interior
36 for Force transducer constraint
37 for Force transducer penalty
38 for Forming node to surface
39 for Forming one way surface to surface
40 for Forming surface to surface
41 for Discrete nodes impacting surface w/ interference
42 for One way treatment of sliding, impact, friction w/ interference
43 for Spotweld
44 for Spotweld with torsion
45 for Sliding, impact, friction w/ interference

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

46 for Tiebreak nodes only
47 for Tied with failure
rebar for rebar in concrete 1D sliding

and an *option* can be:

lcrsgo *load_curve_#* for optional load curve defining the resisting stress vs. gap opening

isrch *flag* small penetration in contact search

where *flag* can be

0 for check is off
1 for check is on
2 for check is on, shortest diagonal used

visdam *percent* for viscous damping coefficient in percent of critical

kpf *flag* kinematic partition factor for constraint

where *flag* can be

0 for fully automatic treatment
1 for one way treatment with slave nodes constrained to master surface
-1 for one way treatment with master nodes constrained to slave surface

lcair *load_curve_#* load curve defining airbag thickness

penmax *penetration* maximum penetration

thkopt f*lag* thickness option

where *flag* can be

0 for default from the control cards
1 for thickness is not considered
2 for thickness is considered but rigid bodies are excluded
3 for thickness is considered including rigid bodies
4 for thickness effects are not included

lcfpb *load_curve_#* force vs. penetration behavior load curve

fcm *flag* force calculation method

where *flag* can be

1 for total normal force on surface vs. max. penetration of any node
2 for normal force on each node vs. penetration

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

of node through the surface

3 for normal pressure vs. penetration of node into surface

4 for total normal force vs. max. soft penetration

unstf unloading stiffness

lcbcrf *load_curve_#* load curve giving the bending component of the retaining force

lcnflen *load_curve_#* load curve giving the normal force per unit draw bead length as a function of displacement

dbd *depth* draw bead depth

scllc *factor* scale factor for load curve

nitdb #*_iterations* number of integration points laong the draw bead

slvmat *material_list* ; automatic slave segment materials

sypl slave, do not include faces with normal boundary constraints

serin slave erosion/interior node option

sadjmat slave storage is allocated so that eroding contact can occur

scoufsf *factor* oulomb friction scale factor

svfsf *factor* viscous friction scale factor

snffs *stress_or_force* slave normal stress at failure

ssffs *stress_or_force* slave shear stress at failure

senf *exponent* slave exponent for normal force

sesf *exponent* slave exponent for shear force

mstmat *material_list* ; automatic master segment materials

mypl master, do not include faces with normal boundary constraints

merin master erosion/interior node option

madjmat master storage is allocated so that eroding contact can occur

mcoufsf *factor*

mvfsf *factor*

mnffs *stress_or_force* master normal stress at failure

msffs *stress_or_force* master shear stress at failure

menf *exponent* master exponent for normal force

mesf *exponent* master exponent for shear force

scoef *coefficient* static coefficient of friction

dcoef *coefficient* dynamic coefficient of friction

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

decay *coefficient* dynamic decay coefficient
incslv include slave side in printed and binary force interface file
incmst include master side in printed and binary force interface file
sfsps *factor* scale factor on default slave penalty stiffness
sfmps *factor* scale factor on default master penalty stiffness
vfcoef *coefficient* coefficient for viscous friction
thss *thickness* optional thickness for slave surface
thms *thickness* optional thickness for master surface
sthss *factor* scale factor for slave surface thickness
sthms *factor* scale factor for master surface thickness
btime *time* birth time
dtime *time* death time
softc *flag* soft constraint option

where

flag can be:

0 for penalty formulation

1 for soft constraint formulation

ssoftc *factor* scale factor for constraint forces of soft constraint option
maxpcss *coordinate* maximum parametric coordinate in segment search
srchdp *depth* search depth in automatic contact
ncybs *cycles* number of cycles between bucket sorts
ncyup *cycles* number of cycles between contact force updates for penalty formulations
diskoc disable logic in thickness offset contact to avoid shooting nodes
concon *conductance* contact conductance
radcon *conductance* radiation conductance
gapcs *size* gap critical size
ctofst
atbo *args*

where *args* can be

1 for Slave nodes in contact and which come into contact will permanently stick. Tangential motion is inhibited.

2 normal_stress shear_stress
for Tiebreak is active for nodes which are

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

initially in contact. Until failure, tangential motion is inhibited.

3 normal_stress shear_stress
for Same as 1st option but with failure after sticking

4 for Tiebreak is active for nodes which are initially in contact but tangential motion with frictional sliding is permitted.

5 plastic_stress load_curve
for Tiebreak is active for nodes which are initially in contact. Damage is defined by a load curve.

6 distance
for Tiebreak is active for brick and thick shell nodes which are initially in contact. damage is a linear function between points.

lcid1 *load_curve*	load curve dynamic interface stiffness
lcid2 *load_curve*	load curve transient interface stiffness
isym *option_#*	symmetric plane option
i2d3d *option_#*	segment searching option
sldthk *thickness*	solid element thickness
sldstf *thickness*	solid element stiffness
igap *option_#*	flag implicit convergence behavior
ignore *option_#*	ignore initial penetration in automatic interfaces
edge *distance*	edge to edge penetration check
rbrad *radius*	
rbstr *strength*	
rbshr *modulus*	
rbumax *strain*	
rbexp *exponent*	

Remarks

Sliding interfaces or contact surfaces are constructed in 3 steps. These steps can be done in any order.

1. define the properties
2. select the slave side
3. select the master side, if applicable

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

The **sid** command is used to define the properties. The **si** and **sii** commands are used in the part phase or the merge phase to select the nodes or faces that form the master and slave sides of the interface.

The options for LS-DYNA are large and unique so they have been singled out and fall under the **lsdsi** option of the **sid** command. Some of these sliding interfaces require nodes on the slave side while others require only a set of faces on the slave side. This definition is required so that the proper data can be written to the output file.

With some simulation code formats, one can construct a node set or a face set. This will be written to the output file as a set. Then it is a simple matter to add the keyword command to the output file using a text editor to transform that set into a contact surface or sliding interface. This approach has the problem that nodes may be merged across the two sides because they are not defined as sliding interfaces.

When nodes are merged, nodes across a sliding interface will not be merged. When a merge command is first issued in the merge phase, a table is written listing the number of nodes and faces associated with each sliding interface.

The **dummy** type interface is actually used to avoid merging of nodes. A sliding interface of this type is not written to the output file.

The nodes and faces of a sliding interface or contact surface can be viewed in the merge phase using the **si** option of the **co** command.

If the output option has been selected prior to using the dialogue box to make a selection, only the options available to that output option will be displayed in the dialogue box.

Examples

```
sid 1 tied pnlt 10;
sid 10 rebar rbrad .01;;
sid 13 lsdsi 9 lcrsgo 1 visdam .34 isrch 1 ; ; ;
sid 54 dummy;
```

siinfo list sliding interface definitions

siinfo *<no arguements>*

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

8. Elements

Beam elements can be created in three ways. The primary method of beam element generation extracts the needed nodes from an existing shell of brick part. This is only available within the block or cylinder part phase. The **ibm** and **ibmi** commands create beams along i-lines of the mesh, the **jbm** and **jbmi** along j-lines, and the **kbm** and **kbmi** along k-lines. This is a way to embed beam elements within a shell of brick structure. Alternatively, the material of the parent shell of brick part can be set to 0 so that the part can be generated as usual, but so that the shell or brick elements will not be saved when the part generation is ended. The nodes that are used in any of these beam commands will be saved along with the beam elements.

The second method of beam element generation uses the **bm** command. This is fully interactive. Beams are strung along a 3D curve.

The **beam** and **cbeam** parts create beams by connecting a pair of nodes with a line of beam elements. This is an old method included to be backwards compatible, but it is not interactive.

Beam cross section properties are defined using **bsd** and **bind**. Shell cross section properties are defined with the **sind** command.

Elements and nodes can have their numbers offset using the **offset** command. This makes it easier to glue several models together by concatenating two files.

Cross sectional properties, and in particular thicknesses, are not scaled by the **xsca**, **ysca**, **zsca**, and **csca** commands.

bsd global beam cross section definition

bsd *option_list ;*

where

option_list consists of some of the following:

sthi *thickness*	for both thicknesses in s-direction
sthi1 *thickness*	for first thickness in s-direction
sthi2 *thickness*	for second thickness in s-direction
tthi *thickness*	for both thicknesses in t-direction
tthi1 *thickness*	for first thickness in t-direction
tthi2 thickness	for second thickness in t-direction
ldp *distance*	for NEUTRAL file beams

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

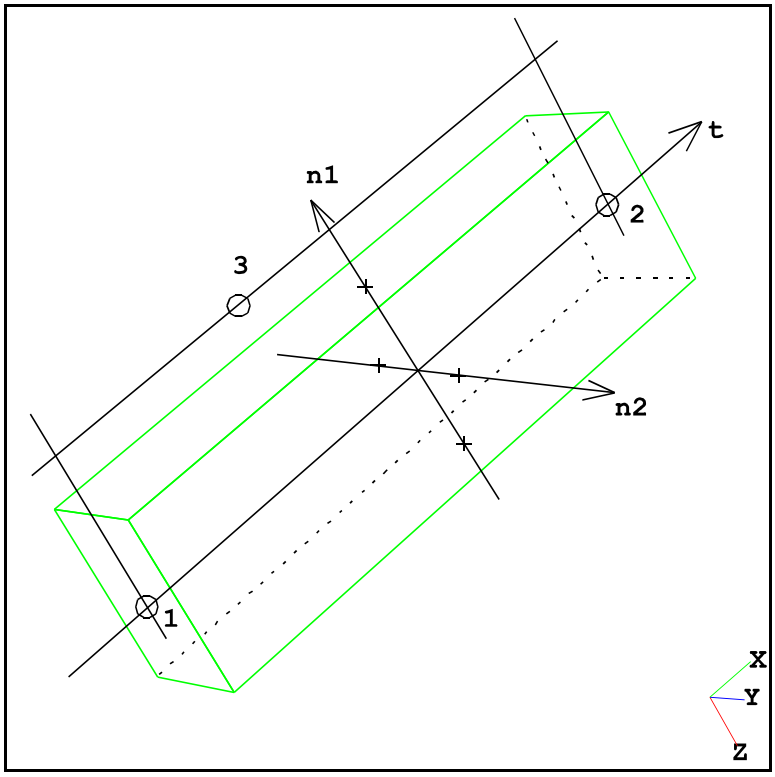


Figure 276 Beam Local Coordinate System for ABAQUS

cstype *type t_options* ;
where
type and *t_options* can be:
7 *t_options ;* for PIPE (ABAQUS)
where
t_options can be
ABCS1 *radius*
ABCS2 *thickness*
NABIP1 *#_integrations*
TRSS *stiffness*
ABTEMP *Temperature*
8 *t_options ;* for BOX (ABAQUS)
where
t_options can be
ABCS1 *width*

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

ABCS2 *height*
ABCS3 *thickness*
ABCS4 *thickness*
ABCS5 *thickness*
ABCS6 *thickness*
NABIP1 *#_integrations*
NABIP2 *#_integrations*
TRSS *stiffness*
ABTEMP *Temperature*

9 *t_options ;* for CIRCLE (ABAQUS)

where

t_options can be

ABCS1 *radius*
NABIP1 *#_integrations*
NABIP2 *#_integrations*
TRSS *stiffness*
ABTEMP *temperature*

10 *t_options ;* for I-BEAM (ABAQUS)

where

t_options can be

ABCS1 *depth*
ABCS2 *height*
ABCS3 *width*
ABCS4 *width*
ABCS5 *thickness*
ABCS6 *thickness*
ABCS7 *thickness*
NABIP1 *#_integrations*
NABIP3 *#_integrations*
TRSS *stiffness*
ABTEMP *temperature*

11 *t_options ;* for RECTANGLE (ABAQUS)

where

t_options can be

ABCS1 *width*
ABCS2 *height*
NABIP1 *#_integrations*
NABIP2 *#_integrations*
TRSS *stiffness*
ABTEMP *temperature*

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

12 *t_options ;* for HEXAGON (ABAQUS)

where

t_options can be

ABCS1 *thickness*
ABCS2 *thickness*
NABIP1 *#_integrations*
NABIP2 *#_integrations*
TRSS *stiffness*
ABTEMP *temperature*

13 *t_options ;* for ELBOW (ABAQUS)

where

t_options can be

ABCS1 *radius*
ABCS2 *thickness*
ABCS3 *radius*
NABIP1 *#_integrations*
NABIP2 *#_integrations*
NABIP3 *#_integrations*
TRSS *stiffness*
ABTEMP *temperature*

14 *t_options ;* for TRAPEZOID (ABAQUS)

where

t_options can be

ABCS1 *width*
ABCS2 *height*
ABCS3 *width*
ABCS4 *depth*
NABIP1 *#_integrations*
NABIP2 #_integrations
RSS *stiffness*
ABTEMP *temperature*

15 *t_options ;* for I-SECTION (ABAQUS)

where

t_options can be

ABCS1 *width*
ABCS2 *height*
ABCS3 *thickness*
ABCS4 *thickness*
NABIP1 *#_integrations*

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

NABIP2 *#_integrations*
TRSS *stiffness*
ABTEMP *temperature*

16 *t_options ;* for ARBITRARY (ABAQUS)

where

t_options can be

CSCRV *y1 z1 ... yn zn* ;
CSSTH thick1 ... *thickn* ;
TRSS *stiffness*
ABTEMP *temperature*

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

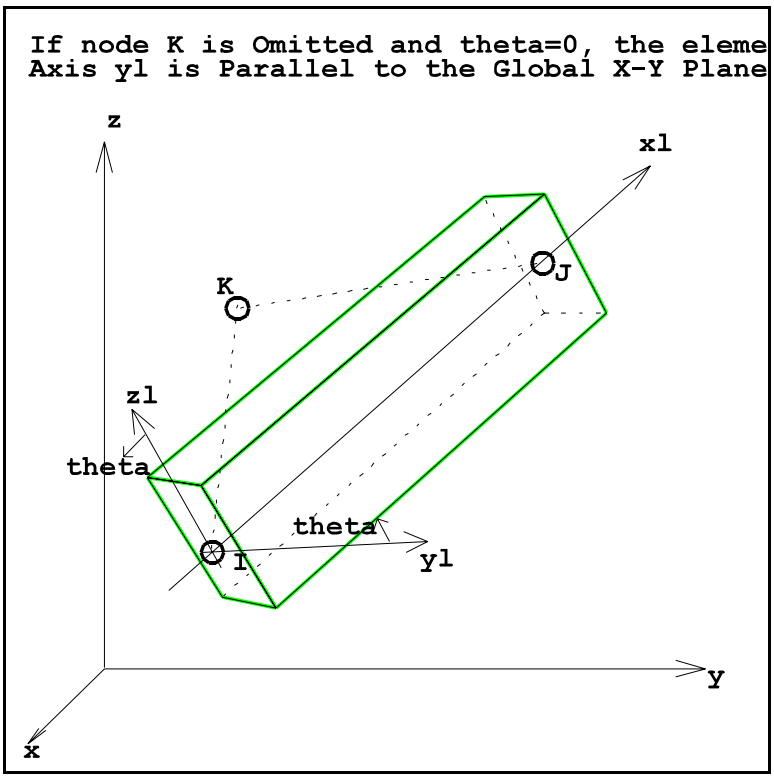


Figure 277 Beam Local Coordinate System for ANSYS

ban4 *t_options* ; for ELASTIC BEAMS (ANSYS)

where

t_options can be

AREA *area*
IXX moment
IYY *moment*
IZZ *moment*
HEIGHT *height*
WIDTH *width*
THETA *angle*
INSTR *strain*
SHEARY *constant*
SHEARZ *constant*

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

ban8 *t_options* ; for SPARS (ANSYS)
where
t_options can be
AREA *area*
INSTR *strain*
ban10 *t_options* ; for TENSION/COMPRESSION SPARS (ANSYS)
where
t_options can be
AREA *area*
INSTR *strain*

ban24 *t_options* ; for THIN WALLED PLASTIC BEAMS (ANSYS)
where
t_options can be
CSCRV *y1 z1 ... yn zn* ;
CSSTH *thick1 ... thickn* ;
RXOFF1 *x-offset*
RXOFF2 *x-offset*
SHEARY *constant*
SHEARZ *constant*
ban33 *t_options* ; for THERMAL BARS (ANSYS)
where
t_options can be
AREA *area*
ban44 *t_options* ; for TAPPERED UNSYMMETRICAL BEAMS (ANSYS)
where
t_options can be
AREA *area*
AREA1 *area*
AREA2 *area*
IXX *moment*
IXX1 *moment*
IXX2 *moment*
IYY *moment*
IYY1 *moment*
IYY2 *moment*
IZZ *moment*
IZZ1 *moment*
IZZ2 *moment*
YB *width*

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

YB1 *width*
YB2 *width*
YA *width*
YA1 *width*
YA2 *width*
ZB *height*
ZB1 *height*
ZB2 *height*
ZA *height*
ZA1 *height*
ZA2 *height*
XOFF1 *x-component*
YOFF1 *y-component*
ZOFF1 *z-component*
XOFF2 *x-component*
YOFF2 *y-component*
ZOFF2 *z-component*
SHEARY *constant*
SHEARZ *constant*

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

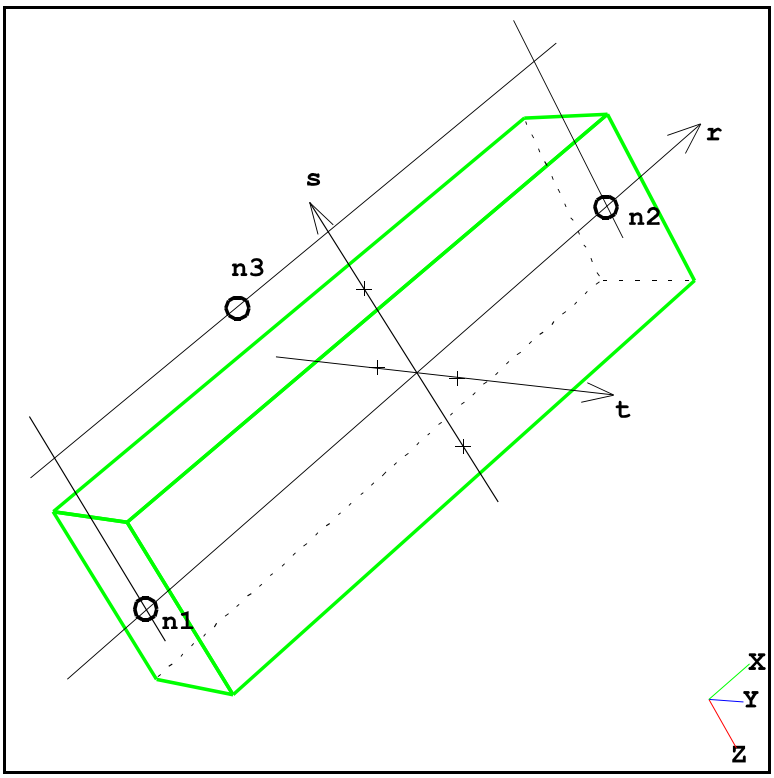


Figure 278 Beam Local Coordinate System for DYNA3D

Parameters

where

parameters can be

(Hughes-Liu beam, constant thickness)

STHI *thickness* (s-thickness at both ends)

TTHI *thickness* (t-thickness at both ends)

or

(Hughes-Liu beam, variable thickness)

STHI1 *thickness* (s-thickness at beginning)

STHI2 *thickness* (s-thickness at ending)

TTHI1 *thickness* (t-thickness at beginning)

TTHI2 *thickness* (t-thickness at ending)

or

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

(Belytschko-Schwer beam)
carea *area* cross section area
iss *iss* area moment of inertia about s-axis
itt *itt* area moment of inertia about t-axis
irr *irr* area moment of inertia about r-axis
sarea *area* shear area of cross section
or
(Truss)
carea *area* cross section area

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

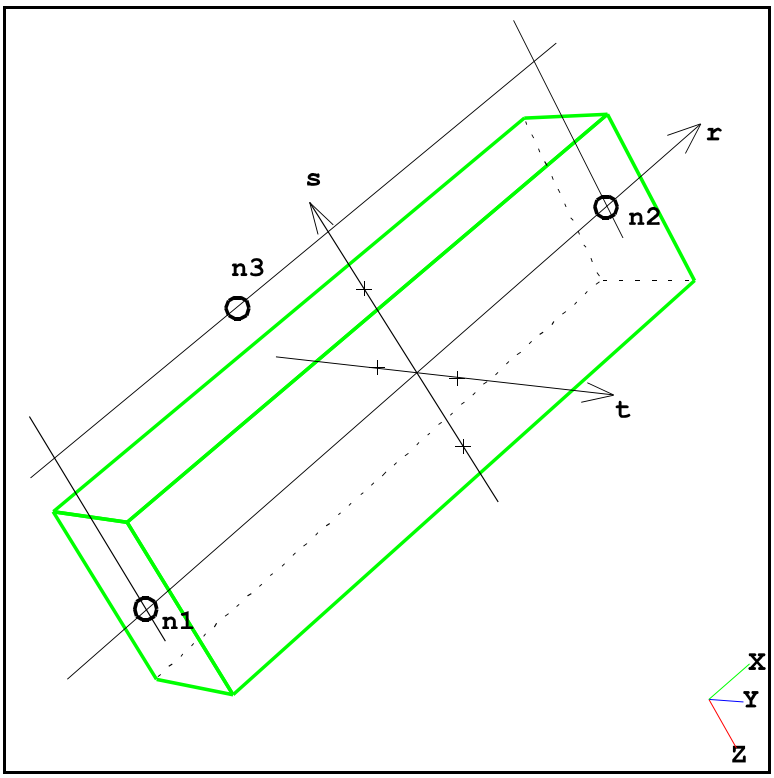


Figure 279 Beam Local Coordinate System for LS-DYNA

Parameters ; for LS-DYNA beams

(Dimensions of Standard Cross Sections are in Global Coordinates)

where

parameters can be either:

(Hughes-Liu Standard Sections)

lsd 1 *flange_width flange_thickness depth web_thickness* (W-section)

lsd 2 *flange_width flange_thickness depth web_thickness* (C-section)

lsd 3 *flange_width flange_thickness depth web_thickness* (angle)

lsd 4 *flange_width flange_thickness depth web_thickness* (T-section)

lsd 5 *flange_width flange_thickness depth web_thickness* (rectangular)

lsd 6 *flange_width flange_thickness depth web_thickness* (Z-section)

lsd 7 *flange_width depth web_thickness* (trapezoidal)

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

SREF *location*
where
location can be:
1 meaning the side where s is 1
0 meaning centered
-1 meaning the side where s is -1
TREF *location*
where
location can be:
1 meaning the side where t is 1
0 meaning centered
-1 meaning the side where t is -1
or
(Hughes-Liu Constant Thickness or Diameter)
STHI *thickness* (s-thickness or outer diameter at both ends)
TTHI *thickness* (t-thickness or inner diameter at both ends)
SREF *location*
where
location can be:
1 meaning the side where s is 1
0 meaning centered
-1 meaning the side where s is -1
TREF *location*
where
location can be:
1 meaning the side where t is 1
0 meaning centered
-1 meaning the side where t is -1
or
(Hughes-Liu Variable Thicknesses or Diameters)
STHI1 *thickness* (s-thickness or outer diameter at beginning)
STHI2 *thickness* (s-thickness or outer diameter at ending)
TTHI1 *thickness* (t-thickness or inner diameter at beginning)
TTHI2 *thickness* (t-thickness or inner diameter at ending)
SREF *location*
where
location can be:
1 meaning the side where s is 1
0 meaning centered
-1 meaning the side where s is -1

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

TREF *location*

where

location can be:

1 meaning the side where t is 1

0 meaning centered

-1 meaning the side where t is -1

or

(Belytschko-Schwer beam)

carea *area* cross section area

iss *iss* area moment of inertia about s-axis

itt *itt* area moment of inertia about t-axis

irr *irr* area moment of inertia about r-axis

sarea *area* shear area of cross section

or

(Truss)

carea *area* cross section area

or

(Belytschko-Schwer Full Integration Beam Standart Sections)

lsd 1 *flange_width flange_thickness depth web_thickness* (W-section)

lsd 2 *flange_width flange_thickness depth web_thickness* (C-section)

lsd 3 *flange_width flange_thickness depth web_thickness* (angle)

lsd 4 *flange_width flange_thickness depth web_thickness* (T-section)

lsd 5 *flange_width flange_thickness depth web_thickness* (rectangular)

lsd 6 *flange_width flange_thickness depth web_thickness* (Z-section)

lsd 7 *flange_width depth web_thickness* (trapezoidal)

or

(Belytschko-Schwer Full Integration Beam Constant Thickness or Diameters)

STHI *thickness* (s-thickness or outer diameter at both ends)

TTHI *thickness* (t-thickness or inner diameter at both ends)

or

(Belytschko-Schwer Full Integration Beam Variable Thicknesses or Diameters)

STHI1 *thickness* (s-thickness or outer diameter at beginning)

STHI2 *thickness* (s-thickness or outer diameter at ending)

TTHI1 *thickness* (t-thickness or inner diameter at beginning)

TTHI2 *thickness* (t-thickness or inner diameter at ending)

or

(Belytschko-Schwer Tubular Beam Constant Diameter)

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

STHI *outer_diameter* (outer diameter at both ends)
TTH *inner_diameter* (inner diameter at both ends)
or
(Belytschko-Schwer Tubular Beam Variable Diameter)
STHI1 *first_outer_diameter* (outer diameter at beginning)
STHI2 *last_outer_diameter* (outer diameter at ending)
TTHI1 *first_inner_diameter* (inner diameter at beginning)
TTHI2 *last_inner_diameter* (innter diameter at ending)
or
(Discrete 3D Beam)
vold *volume*
cabarea *cable_area*
lump *inertia* (lumped geometric inertia)
cablcid *local_coordinate_system_#* (defined by the **lsys** command)
caboff *cable_offset*
or
(Spot Weld Beam Standart Sections)
lsd 1 *flange_width flange_thickness depth web_thickness* (W-section)
lsd 2 *flange_width flange_thickness depth web_thickness* (C-section)
lsd 3 *flange_width flange_thickness depth web_thickness* (angle)
lsd 4 *flange_width flange_thickness depth web_thickness* (T-section)
lsd 5 *flange_width flange_thickness depth web_thickness* (rectangular)
lsd 6 *flange_width flange_thickness depth web_thickness* (Z-section)
lsd 7 *flange_width depth web_thickness* (trapezoidal)
or
(Spot Weld Beam Constant Thickness)
STHI *thickness* (s-thickness at both ends)
TTHI *thickness* (t-thickness at both ends)
or
(Spot Weld Beam Variable Thicknesses)
STHI1 *thickness* (s-thickness at beginning)
STHI2 *thickness* (s-thickness at ending)
TTHI1 *thickness* (t-thickness at beginning)
TTHI2 *thickness* (t-thickness at ending)

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

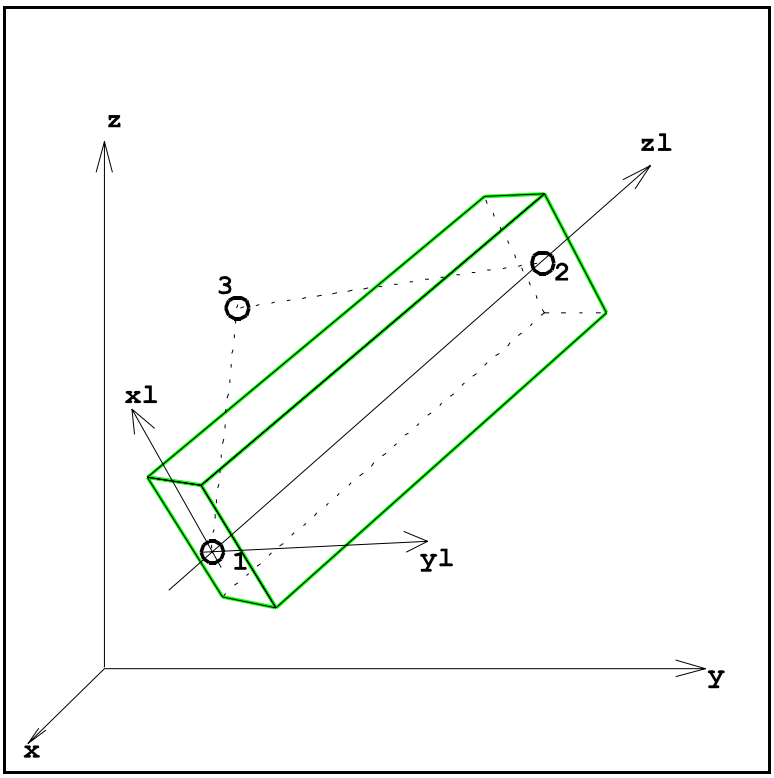


Figure 280 Beam Local Coordinate System for MARC

marc13 *t_options* ; for OPEN SECTION THIN WALLED BEAMS (MARC)
where
t_options can be:
CSCRV *y(1) z(1) ... y(n+1) z(n+1) ;*
CSSTH *thick(1) ... thickn* ;
VCSSTH *first_thick(1) last_thick(1)... first_thick(n) last_thick(n)* ;
CSSLEN *length(1) ... length(n)* ;
CSSSLP *first_dy(1) first_dz(1) last_dy(1) last_dz(1) ... first_dy(n) first_dz(n) last_dy(n) last_dz(n) ;*
CSDIV *#_divisions(1) ... #_divisions(n)* ;

marc14 *t_options* ; for THIN WALLED BEAMS W/O WARPING

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

(MARC)

where

t_options can be:

CSCRV *y(1) z(1) ... y(n+1) z(n+1) ;*

CSSTH *thick(1) ... thickn ;*

VCSSTH *first_thick(1) last_thick(1) ...first_thick(n) last_thick(n) ;*

CSSLEN *length(1) ... length(n)* ;

CSSSLP *first_dy(1) first_dz(1) last_dy(1) last_dz(1) ... first_dy(n) first_dz(n) last_dy(n) last_dz(n)* ;

CSDIV *#_divisions(1) ... #_divisions(n)* ;

THICK *thickness*

RADIUS *radius*

marc25 *t_options* ; for THIN WALLED BEAMS (MARC)

where

t_options can be:

CSCRV *y(1) z(1) ... y(n+1) z(n+1) ;*

CSSTH *thick(1) ... thickn* ;

VCSSTH *first_thick(1) last_thick(1) ...first_thick(n) last_thick(n)* ;

CSSLEN *length(1) ... length(n)* ;

CSSSLP *first_dy(1) first_dz(1) last_dy(1) last_dz(1) ... first_dy(n) first_dz(n) last_dy(n) last_dz(n)* ;

CSDIV *#_divisions(1) ... #_divisions(n)* ;

THICK *thickness*

RADIUS *radius*

marc31 *t_options* ; for ELASTIC CURVED PIPE (MARC)

where

t_options can be:

AREA *area*

IYY *moment*

IZZ *moment*

IXX *moment*

ETSAY *area*

ETSAZ *area*

THICK *thickness*

RADIUS *radius*

BEND *radius*

marc52 *t_options* ; for ELASTIC BEAMS (MARC)

where

t_options can be

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

AREA *area*
IYY *moment*
IZZ *moment*
IXX *moment*
ETSAY *area*
ETSAZ *area*

marc76 *t_options* ; for THIN WALLED BEAMS W/O WARPING (MARC)

where

t_options can be:

CSCRV *y(1)* z(1) *... y(n+1) z(n+1)* ;
CSSTH *thick(1) ... thickn* ;
VCSSTH *first_thick(1) last_thick(1)... first_thick(n) last_thick(n)* ;
CSSLEN *length(1) ... length(n)* ;
CSSSLP *first_dy(1) first_dz(1) last_dy(1) last_dz(1) ... first_dy(n) first_dz(n) last_dy(n) last_dz(n)* ;
CSDIV *#_divisions(1) ... #_divisions(n)* ;
THICK *thickness*
RADIUS *radius*

marc79 *t_options* ; for THIN WALLED BEAMS WITH WARPING (MARC)

where

t_options can be:

CSCRV *y(1) z(1) ... y(n+1) z(n+1)* ;
CSSTH *thick(1) ... thickn* ;
VCSSTH *first_thick(1) last_thick(1)... first_thick(n) last_thick(n)* ;
CSSLEN *length(1) ... length(n)* ;
CSSSLP *first_dy(1) first_dz(1) last_dy(1) last_dz(1) ... first_dy(n) first_dz(n) last_dy(n) last_dz(n)* ;
CSDIV *#_divisions(1) ... #_divisions(n)* ;

marc98 *t_options* ; for ELASTIC BEAMS WITH TRANSVERSE SHEAR (MARC)

where

t_options can be:

AREA *area*
IYY *moment*
IZZ *moment*
IXX *moment*
ETSAY *area*
ETSAZ *area*

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

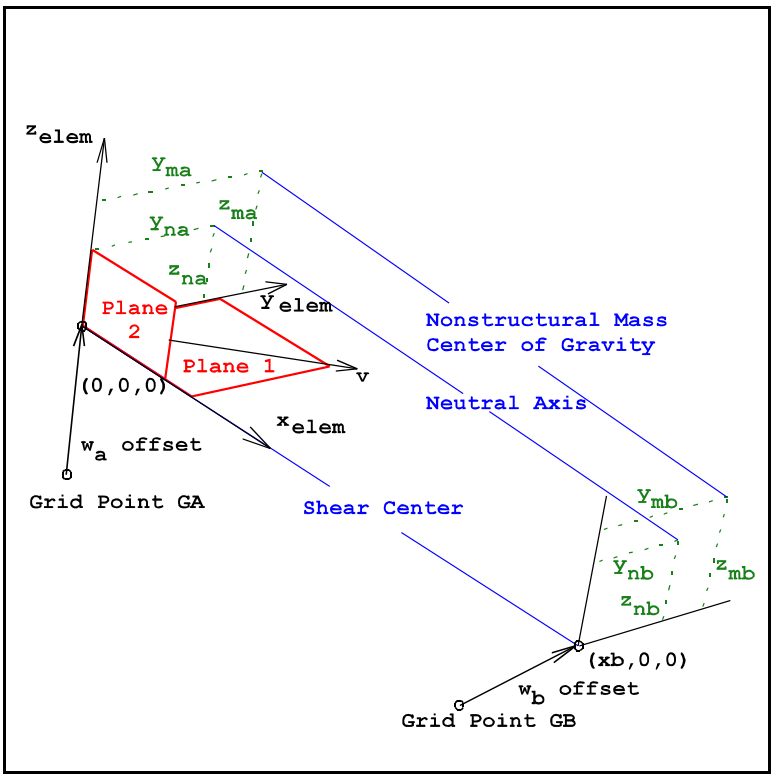


Figure 281 Beam Local Coordinate System for NASTRAN Beam

bna1 *t_options ;*

where

t_options can be:

SHSTF *y-shear* z-shear
SHRLF *y-shear* z-shear
NSMMI *moment*
NSMMI1 *moment*
NSMMI2 *moment*
WARP *coefficient*
WARP1 *coefficient*
WARP2 *coefficient*
CENGRAV *y1 z1 y2 z2*
NEUAXIS *y1 z1 y2 z2*

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

SPCSD *position c_options* ;
where
c_options can be
AREA *area*
IYY *moment*
IZZ *moment*
IYZ *moment*
IXX *moment*
MASS *mass*
SDR1 *y1 z1*
SDR2 *y2 z2*
SDR3 *y3 z3*
SDR4 *y4 z4*
DX1
DZ1
RX1
RY1
RZ1
DX2
DY2
DZ2
RX2
RY2
RZ2

bna2 *t_options* ; for OFFSET RODS (NASTRAN)
where
t_options can be:
AREA *area*
IYY *moment*
IZZ *moment*
IYZ *moment*
IXX *moment*
MASS *mass*
CENGRAV y *z*
NEUAXIS y *z*
CSTYPE *type c_options* ;
where
type can be
1 for the default elliptic
2 for symmetry about y and z

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

3 for symmetry about y
4 for symmetry about z
5 for symmetry about y=z=0
6 for arbitrary

c_options can be

CSCRV *y1 z1 ... yn zn* ;
CSSAR *area1 ... arean* ;
CSSMAT *material1 ... materialn* ;

DX1
DY1
DZ1
RX1
RY1
RZ1
DX2
DY2
DZ2
RX2
RY2
RZ2

bna3 *t_options* ; for CURVED BEAMS (NASTRAN)

where

t_options can be:

AREA *area*
IYY *moment*
IZZ *moment*
IXX *moment*
RB *radius*
THETAB *angle*
SHSTF *y-shear z-shear*
SDR1 *y1 z1*
SDR2 *y2 z2*
SDR3 *y3 z3*
SDR4 *y4 z4*
RC *offset*
ZC *offset*
DELTAN *offset*
DX1
DY1
DZ1

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

RX1
RY1
RZ1
DX2
DY2
DZ2
RX2
RY2
RZ2

bna4 *t_options* ; for ELBOW & CURVED PIPE (NASTRAN)

where

t_options can be:

FSI *intensification*
MCSR *radius*
WTH *thickness*
IP *pressure*
RB *radius*
THETAB *angle*
MASS *mass*
RC *offset*
ZC *offset*
DX1
DY1
DZ1
RX1
RY1
RZ1
DX2
DY2
DZ2
RX2
RY2
RZ2

bna5 *t_options* ; for SIMPLE BEAM (BAR) (NASTRAN)

where

t_options can be:

AREA *area*
IYY *moment*
IZZ *moment*

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

IYZ *moment*
IXX *moment*
MASS *mass*
SHSTF *y-shear* z-shear
SDR1 *y1* z1
SDR2 *y2* z2
SDR3 *y3* z3
SDR4 *y4* z4
DX1
DY1
DZ1
RX1
RY1
RZ1
DX2
DY2
DZ2
RX2
RY2
RZ2

bna6 *t_options* ; for ROD (NASTRAN)

where

t_options can be:

AREA *area*
IXX *moment*
TSC *coefficient*
MASS *mass*
DX1
DY1
DZ1
RX1
RY1
RZ1
DX2
DY2
DZ2
RX2
RY2
RZ2

bna7 *t_options* ; for TUBE (NASTRAN)

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

where
t_options can be:
OD *diameter*
WTH *thickness*
MASS *mass*
OD2 *diameter*
DX1
DY1
DZ1
RX1
RY1
RZ1
DX2
DY2
DZ2
RX2
RY2
RZ2

NE/NASTRAN Beams (in addition to all of the beam cross sections for NASTRAN)

bna8 *t_options* ; for CABLE (NE/NASTRAN)
where
t_options can be:
islack *slack*
itens *tension*
area *area*
iyz *moment*
ats *stress*

Remarks

For detailed information on definition of the cross-section see the manual of the specific simulation code.

If the output option has been selected prior to using the dialogue box to make a selection, only the options available to that output option will be displayed in the dialogue box.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

bsinfo write information about defined beam cross sections

bsinfo *bsd_#*

where

bsd_# is the one assigned to the definition in the **bsd** command.

bind Hughes-Liu beam user-defined integration points

bind *rule_#* $s_1\ t_1\ w_1\ s_2\ t_2\ w_2\ s_3\ t_3\ w_3$ ... ;

where

rule_# can be any positive integer used to refer to this rule followed by a list of local coordinates of integration points s_i and t_i and corresponding weights w_i .

Remarks

This command is used to define the integration points for the Hughes-Liu beam in **DYNA3D** and **LS-DYNA**. The coordinates s_i and t_i are parametric coordinates of integration points from interval <-1,1> (Denoted by crosses). The weights w_i are determined from the term:

$$w_i = A_i / A$$

where A_i is the area corresponding to the *i-th* integration node. A is the total area of the cross section determined by:

$$A = \sum A_i.$$

The t_t and s_t dimensions are used for scaling from parametric to real coordinates. The t_t and s_t dimensions are specified using the **bsd**, **bm**, **ibm**, **ibmi**, **jbm**, **jbmi**, **kbm**, **kbmi**, **dynamats**, or **lsdymats** commands.

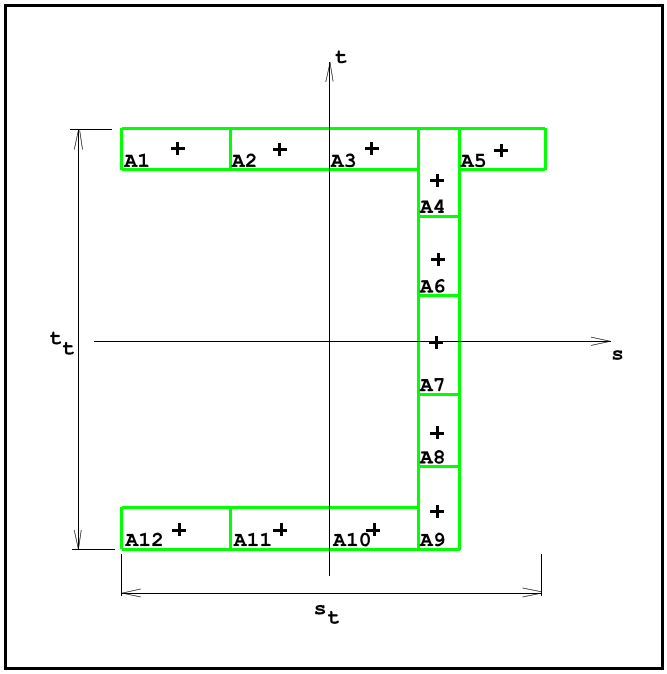


Figure 282 Cross Section with Integration Points

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

lsbsd list the defined beam cross sections.

lsbsd

Remarks

Beam cross section properties are listed by number and type. Use **bsinfo** to get full information about a specific beam cross section definition.

offset add offset to numbered entities in the output

offset *type offset*

where *type* can be

nodes	node numbers
bricks	brick elements (or all elements)
shells	shell elements (if numbered independently)
beams	beam elements (if numbered independently)
tshells	thick shells
nsetoff	node sets if they are automatically numbered (not named)
fsetoff	face sets if they are automatically numbered (not named)
esetoff	element sets if they are automatically numbered (not named)
partoff	parts
lcrsyoff	local coordinate systems

Remarks

Only keyword outputs can use this number offset feature. For example, **Ls-dyna**, **Abaqus**, **Ansys**, **Marc**, **Nastran**, **NE/Nastran**, and **Neutral**.

Abaqus uses **nodes**, **bricks**, **nsetoff**, **esetoff**, **partoff,** and **nsetoff** affects the automatically numbered node sets as a result of the **fc**, **fd**, **fv**, **ft**, **acc**, and **mom** nodal boundary conditions. **Esetoff** affects the automatically numbered element sets as a result of the **pr** condition. **Partoff** affects the automatic numbering of element sets based on the part number.

Ansys uses the **bricks** and **nodes** offsets.

Lsdyna uses all of the offsets.

Marc uses the **bricks** and **nodes** offsets.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Nastran and **NE/Nastran** uses the **bricks**, **nodes**, **nsetoff**, and **esetoff** offsets.

Neutral uses the **bricks** and **nodes** offsets.

If the output option has been selected prior to using the dialogue box to make a selection, only the options available to that output option will be displayed in the dialogue box.

sind shell user-defined integration rules

sind *rule* s_1 t_1 w_1 s_2 t_2 w_2 ... s_n t_n w_n ;

where

rule	is the integration rule number, 0 or 1. If 1, then all other arguments are ignored.
t_i w_i m_i	are local coordinate, integration weight, and material number

9. Sets

These commands are found in all phases. Named sets are a useful tool in the definition of boundary conditions and properties in the mesh and form an alternative to selecting regions in the Part phase. An arbitrary selection can be made and this is the advantage to using set functions in the Merge phase. The disadvantage is that the selection may no longer be parametric. If you go back and make a change to the mesh, you may have to redefine the set since the element and node numbers have changed.

The name of the set can be up to 8 alphanumeric characters long. Each name of the set must be unique.

In some of the set commands, the logical or Boolean set operators AND and OR are used to create new sets from existing sets. The AND operator between two sets means to take their intersection. This should not be confused with the common usage of and which might be interpreted to mean the addition of two sets. The OR operator does this function.

delset delete a set

delset *type set_name*

where *type* is the type of set which can be one of the following

node	node set

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

face face set
element element set

where *set_name* is the name of the set.

Remarks

If a node set was constructed but is no longer needed, then it is best to delete it with this command. This can be important if an output file is going to be written which automatically writes all sets. When deleted, the set will not be written to the output file. In addition, it will not be using memory. This deletion has no affect on the nodes and elements of the mesh. All that is deleted is the list of nodes, faces, or elements that formed the set.

esetc element set comment

esetc *set_name text*

where

set_name is the name of the element set
text a text comment

Remarks

It is necessary to specify a comment using **esetc** for each node set to be written to ALE3D.

esetinfo report the element set names and number of elements

esetinfo (no arguments)

Remarks

Command **esetinfo** reports the element set names and number of elements.

fsetc face set comment

fsetc *set_name text*

where

set_name is the name of the face set
text a text comment

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

It is necessary to specify a comment using **fsetc** for each node set to be written to ALE3D.

fsetinfo report the face set names and number of faces

fsetinfo (no arguments)

Remarks

Command **fsetinfo** reports the face set names and number of faces.

nsetc attach a comment to a node set

nsetc *set_name text*

where

set_name	is the name of the node set
text	a text comment

Remarks

It is necessary to specify a comment using **nsetc** for each node to be written to ALE3D.

nsetinfo report the node set names and number of nodes

nsetinfo (no arguments)

Remarks

Command **nsetinfo** reports the node set names and number of nodes.

10. Coordinate Transformations and Part Replication

Coordinate transformations are used to translate, scale, reflect, and rotate an object from a local coordinate system to the global coordinate system. A local coordinate system is a frame of reference in which to generate a part. A local coordinate system is almost always a matter of convenience. The global coordinate system refers to the frame of reference used to create the complete model. In

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

many cases, a part is generated in the global coordinate system, so there is no need to transform it. When you take advantage of symmetry by generating a section of the model, you will need some of the commands in this section to replicate the part.

The **lct** (local coordinate system) command should not be confused with the notion of a local coordinate system and the **gct** (global coordinate transformation) command should not be confused with the notion of a global coordinate system. This unfortunate naming cannot be repaired because such a change would cause many old input files to fail to generate the expected models.

In some cases, a component is duplicated many times. Each duplicate component must be transformed or placed into its proper location within the global coordinate system. The part duplication commands depend on the coordinate system transformations. Coordinate transformations can be defined in the **lct**, **gct**, and **lev** commands. Parts can be replicated with the **lrep**, **grep**, and **pslv** commands which refer back to the transformations defined in the **lct**, **gct**, and **lev** commands, respectively. The **lrep** and **grep** commands are discussed in the part phase section.

Properties are replicated along with the parts, such as material numbers and sliding interfaces (contact surface). Some of these properties are numbered and the number can be increased for each replication. These are replication increment commands found in this section.

In all cases, a coordinate transformation in **True*Grid*®** is composed of a sequence of basic operations. Each basic operation is given by a keyword possibly followed by some parameters. Each basic operation is performed in order from left to right. This ordering of the basic operations is sometimes referred to as a product or composition of basic operations. The composition of the basic operations is referred to as a single coordinate transformation.

It can be difficult to think of a complex transformation in three dimensions. You can simplify this by thinking of the object already in the global coordinate system. Then build the transformation, one operation at a time until you have moved, rotated, reflected, and scaled it to the proper position, orientation, and size.

Some coordinate transformations are applied to all parts, such as **csca**, **xcsa**, **ycsa**, **zcsa**, **xoff**, **yoff**, **zoff**, **exch**, and **gexch**. These transformations are referred to as final transformations because they are the last transformations that are applied to the parts. These commands are usually issued at the beginning of a command file to change dimensions.

There are four types of transformations that can be applied to a part. They are: local (**lct**), global (**gct**), levels (**lev**), and final transformations and they are applied in the order they are listed. Since the **lev** and the associated **pslv**/**pplv** commands allow for many nested levels of transformations, the ordering is based on the ordering of the nested levels. The inner nested level is performed first, with

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

the outer level performed last. This ordering is like nested loops in a programming language.

Element cross sectional properties, and in particular thicknesses, are not scaled by these transformations.

lct define local coordinate transformations

lct *n* $trans_1$ *; ... ;* $trans_n$ *;*

where $trans_i$ is a left-to-right product of the following basic operations:

- **mx** *x_offset* to translate in the x direction
- **my** *y_offset* to translate in the y direction
- **mz** *z_offset* to translate in the z direction
- **v** *x_offset y_offset z_offset* to translate by a vector
- **rx** *theta* to rotate about the x axis
- **ry** *theta* to rotate about the y axis
- **rz** *theta* to rotate about the z axis
- **raxis** *angle x0 y0 z0 xn yn zn* axis of rotation
- **rxy** to reflect about the x-y plane
- **ryz** to reflect about the y-z plane
- **rzx** to reflect about the z-x plane
- **tf** *origin x-axis y-axis*

  where each of the arguments consist of a coordinate type followed by coordinate information:

  - **rt** *x y z* Cartesian coordinates
  - **cy** *rho theta z* cylindrical coordinates
  - **sp** *rho theta phi* spherical coordinates
  - **pt** *c.i* label of a labeled point from a 3D curve
  - **pt** *s.i.j* label of a labeled point from a surface
- **ftf** *1st_origin 1st_x-axis 1st_y-axis 2nd_origin 2nd_x-axis 2nd_y-axis*

  where each of the arguments consist of a coordinate type followed by coordinate information:

  - **rt** *x y z* Cartesian coordinates
  - **cy** *rho theta z* cylindrical coordinates
  - **sp** *rho theta phi* spherical coordinates
  - **pt** *c.i* label of a labeled point from a 3D curve
  - **pt** *s.i.j* label of a labeled point from a surface
- **inv** invert the present transformation)
- **csca** *scale_factor* to scale all coordinates
- **xsca** *scale_factor* to scale the x coordinate

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

ysca *scale_factor*	to scale the y coordinate
zsca *scale_factor*	to scale the z coordinate
repe *#_repetitions*	to repeat powers of the current transformation
save *transform_#*	to apply a previous transformation
last	to use the last transformation

Remarks

This command defines a sequence of *n* local coordinate transformations; once you issue an **lct** command the results of any previous **lct** commands are no longer available. Each transformation is a product of primitive operators. The product is formed from the left to the right. One copy of the part is created and transformed for each transformation defined in the **lct** command and referenced in the **lrep** command.

You may wish to build a part in a convenient local coordinate system, and then transform the part to the proper position in the global coordinate system. The **lct** command, in conjunction with the **lrep** command, is used to move, scale, reflect, and rotate the part to put it in its place in the model relative to the other parts.

Examples

lct 3 **rz** 45 ; **rz** 45 **mz** 10 ; **rz** 45 **mz** 10 **mx** 25 ;

This command defines 3 local coordinate transformations numbered from 1 to 3. Once you have issued this command, you can no longer reference any previously defined local coordinate transformations.

The first transformation is a rotation about the z-axis. The second transformation is defined in two operations. First the object is rotated about the z-axis 45 degrees. Then the object is moved 10 units in the z-direction. The third transformation is defined by three operations. The first and second operations are the same as the second transformation. In addition, the third operation then moves the object in the x-direction 25 units. This command is equivalent to

lct 3 **rz** 45 ; **last mz** 10 ; **last mx** 25 ;

which is also equivalent to

lct 3 ; **rz** 45 ; **save** 1 **mz** 10 ; **save** 2 **mx** 25 ;

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

gct define global coordinate transformations

gct *n* $trans_1$ *; ... ;* $trans_n$ *;*

where $trans_i$ is one or more of:

- **mx** *x_offset* — to translate in the x direction
- **my** *y_offset* — to translate in the y direction
- **mz** *z_offset* — to translate in the z direction
- **v** *x_offset y_offset z_offset* — to translate by a vector
- **rx** *theta* — to rotate about the x axis
- **ry** *theta* — to rotate about the y axis
- **rz** *theta* — to rotate about the z axis
- **raxis** *angle x0 y0 z0 xn yn zn* — axis of rotation
- **rxy** — to reflect about the x-y plane
- **ryz** — to reflect about the y-z plane
- **rzx** — to reflect about the z-x plane
- **tf** *origin x-axis y-axis*

  where each of the arguments consist of a coordinate type followed by coordinate information:

  - **rt** *x y z* — Cartesian coordinates
  - **cy** *rho theta z* — cylindrical coordinates
  - **sp** *rho theta phi* — spherical coordinates
  - **pt** *c.i* — label of a labeled point from a 3D curve
  - **pt** *s.i.j* — label of a labeled point from a surface
- **ftf** *1st_origin 1st_x-axis 1st_y-axis 2nd_origin 2nd_x-axis 2nd_y-axis*

  where each of the arguments consist of a coordinate type followed by coordinate information:

  - **rt** *x y z* — Cartesian coordinates
  - **cy** *rho theta z* — cylindrical coordinates
  - **sp** *rho theta phi* — spherical coordinates
  - **pt** *c.i* — label of a labeled point from a 3D curve
  - **pt** *s.i.j* — label of a labeled point from a surface
- **inv** — invert the present transformation
- **csca** *scale_factor* — to scale all coordinates
- **xsca** *scale_factor* — to scale the x coordinate
- **ysca** *scale_factor* — to scale the y coordinate
- **zsca** *scale_factor* — to scale the z coordinate
- **repe** *#_repetitions* — to repeat powers of the current transformation
- **save** *transform_#* — to apply a previous transformation
- **last** — to use the last transformation

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

This command defines the entire sequence of global coordinate transformations; once you issue a **gct** command the results of any previous **gct** commands are no longer available. Each transformation is a product of primitive operators. The product is formed from the left to the right.. One copy of the part is created and transformed for each transformation defined in the **gct** command and referenced in the **grep** command.

You may wish to build a part in a convenient local coordinate system, and then transform the part to the proper position in the global coordinate system. The **gct** command, in conjunction with the **grep** command, is used to move, scale, reflect, and rotate the part to put it in its place in the model relative to the other parts.

Examples

```
gct 7 rxy; ryz; rzx; ryz rzx; rxy ryz; rxy rzx; rxy ryz rzx;
```

This set of transformations can be used to reflect a part from the first octant into all of the other 7 octants. The next example shows these transformations in use.

```
gct 7 rxy;ryz;rzx;ryz rzx;
rxy ryz;rxy rzx;rxy ryz rzx;
block 1 2 3;1 2 3 4;1 2 3 4;
     .25   .975  1
     -.2  -.025 .025 .22
     -.2  -.025 .025 .2
dei 1 2;1 2 0 3 4;;
dei 1 2;;1 2 0 3 4;
dei 2 3;1 2 0 3 4;1 2 0 3 4;
mb 2 2 2 3 3 3 x .25
tr 1 1 1 3 4 4 rz 45 ry -45;
grep 0 1 2 3 4 5 6 7;
endpart
```

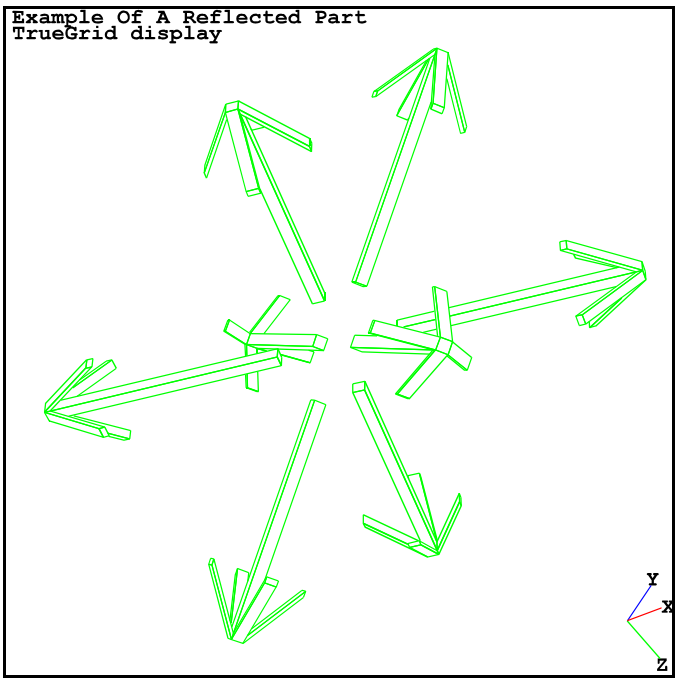


Figure 283 A Part Reflected Into 7 Octants

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

lev define a set of transformations to replicate a set of parts

lev *level_# options ;*

level_# is a small positive number to uniquely identify the level

where an *option* can be any of:

grep *list_global_transform_# ;* a list of global coordinate transformations numbers (see the **gct** command)

add *level_#* include all transforms from another numbered **lev** command

prod *first_level_# second_level_#* to include a product of transforms of two numbered **lev** commands

levct *n* $trans_1$ *;* $trans_2$ *; ... ;* $trans_n$ *;*

where $trans_i$ is one or more of:

mx *x_offset* to translate in the x direction

my *y_offset* to translate in the y direction

mz *z_offset* to translate in the z direction

v *x_offset y_offset z_offset* to translate by a vector

rx *theta* to rotate about the x axis

ry *theta* to rotate about the z axis

rz *theta* to rotate about the z axis

raxis *angle x0 y0 z0 xn yn zn* axis of rotation

rxy to reflect about the x-y plane

ryz to reflect about the y-z plane

rzx to reflect about the z-x plane

tf *origin x-axis y-axis*

where each of the arguments consist of a coordinate type followed by coordinate information:

rt *x y z* Cartesian coordinates

cy *rho theta z* cylindrical coordinates

sp *rho theta phi* spherical coordinates

pt *c.i* label of a labeled point from a 3D curve

pt *s.i.j* label of a labeled point from a surface

ftf *1st_origin 1st_x-axis 1st_y-axis 2nd_origin 2nd_x-axis 2nd_y-axis*

where each of the arguments consist of a coordinate type followed by coordinate information:

rt *x y z* Cartesian coordinates

cy *rho theta z* cylindrical coordinates

sp *rho theta phi* spherical coordinates

pt *c.i* label of a labeled point from a 3D curve

pt *s.i.j* label of a labeled point from a surface

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

inv	invert the present transformation
csca *scale_factor*	to scale all coordinates
xsca *scale_factor*	to scale the x coordinate
ysca *scale_factor*	to scale the y coordinate
zsca *scale_factor*	to scale the z coordinate
repe #_*repetitions*	to repeat powers of the current transformation
save *transform_#*	to apply a previous transformation
last	to use the last transformation

Remarks

This command defines a set of coordinate transformations associated with a level number. After defining a level, you can apply it to several parts at once, with the **pslv** and **pplv** commands. These two commands bracket the set of parts that the level is applied to. **True*Grid*®** will apply the level transformations to every part defined after a **pslv** and before the matching **pplv**. The way you first define and then apply a level is analogous to the way you first define a local or global coordinate transformation, **lct** or **gct**, and then apply them with **lrep** or **grep**.

You can define up to 20 levels of transformations. And each level can have an unlimited number of transformations. Once you define a level, you can reference it with **pslv** to replicate a set of parts. A level of transformations can be referenced by **pslv** commands many times. A level can be redefined.

The **levct** option defines a list of coordinate transformations in the same fashion as the **lct** and **gct** commands. That is, this option defines a transformation as a composition of a number of basic operations.

The **grep** option includes in the level a subset of previously defined global coordinate transformations. You identify them by their sequence numbers in the last **gct** command. A warning message is issued if a referenced global transformation was not previously defined by a **gct** command. Do not confuse this option with the part command **grep**. The **grep** command directly uses global transformations to replicate and transform a part. This **grep** option only uses global transformations to help define the coordinate transformations that constitute the level.

The **add** option will include in the present level definition all of the transformations associated with a previously defined level.

With the **prod** option, you can form the products of all transformations in two previously defined levels, and include all those products as transformations of your new level. This means that for every coordinate transformation in the first level and every coordinate transformation in the second level,

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

True*Grid*® forms the product transformation and includes that in the new level's list of coordinate transformations. A product of two coordinate transformations is defined to be the result of first applying the coordinate transformation from the first level and then applying the coordinate transformation from the second level.

Examples

```
gct 3 mx 10 ; repe 3 ;
lev 1 levct 3 rx 30 ; rx 30 mz 10 ; rx 30 mz 10 my 10 ;;
lev 2 grep 0 2 3 ; ;
lev 3 add 2 levct 1 mx -10 ; grep 1 ; ;
lev 4 prod 1 3 ; ;
```

In the following example, a small part, in the shape of a sphere is created and then replicated using the **lev** command. The picture is below. The point to this example is that the second **lev** is nested in the first **lev** command using the **pslv** and **pplv** pair of commands, much like the way loop statements can be nested in a programming language.

```
title Example Of The LEV Command
gct 1 mx 4 my 4 rz 15;
lev 1 grep 0;
      levct 11 rz 30;repe 11;;
lev 2 grep 0 1;;
pslv 1
pslv 2
block 1 4;1 4;1 4;5 7 5 7 5 7
sfi -1 -2;-1 -2;-1 -2;sp 6 6 6 1
endpart
pplv
pplv
merge
```

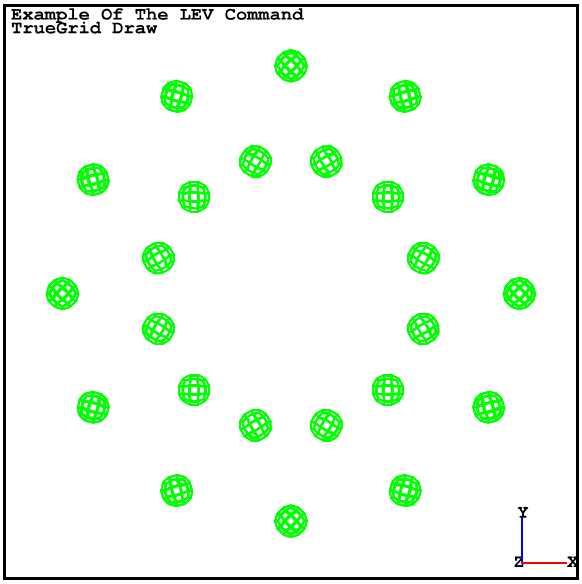


Figure 284 Nested LEV Replications

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

pslv begin replicating multiple parts

pslv *level_#*

where *level_#* is the level definition number assigned in a **lev** command.

Remarks

This command is usually not issued interactively, although there is no compelling reason for this. It usually requires good planning, since it applies to multiple parts. Typically a set of parts are generated and the commands saved in a batch command file for regeneration. Then this command is inserted in the batch command file to produce the desired replications.

Before using the **pslv** and the related **pplv** commands, define a numbered set of transformations with the **lev** command.

The **pslv** and **pplv** commands will replicate a set of parts. Each use of the **pslv** command is paired with the use of a **pplv** command. The **pslv** marks the beginning and **pplv** marks the ending. All parts in between these two commands are replicated, one replication for each transformation found in the corresponding **lev** command. Neither the **pslv** nor the **pplv** command can be issued in the part phase.

For each **pslv** command there must follow somewhere a **pplv** command. The **pslv** and **pplv** pair of commands can be nested. When you nest one **pslv**/**pplv** pair within another, the effect is that of taking the product of the transformations in the two corresponding levels of transformations. A stacking technique is used to handle the products of replications. Each nesting of replications adds another level to the replication stack. The command **pslv** stands for "**p**u**s**h a **l**e**v**el onto the stack" and the command **pplv** stands for "**p**o**p** the top **l**e**v**el off of the stack".

If there are nested **pslv**/**pplv** commands, care is needed to determine the pairing of the **pslv** command with its corresponding **pplv** commands. This pairing is necessary to determine the scope of each **pslv** command (i.e. the parts that are replicated). The best way to determine which **pplv** is paired to which **pslv** command is to use a stacking technique. As you inspect the sequence of parts being generated, mentally place each **pslv** command encountered onto a stack. For each **pplv** that is encountered, pop the top **pslv** off of the stack which is then paired with the **pplv** command.

Pslv/**pplv**s pairs can be nested up to 20 deep. Local or global replication commands (**lrep** or **grep**) are also pushed onto and later popped off of the replication stack as a part is replicated. So when either of these commands are used, the total number of possible nested **pslv**/**pplv**s that can be used is reduced.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Examples

```
 pslv 1
 pslv 2
c part number 1
   block ...
    ...
   endpart
  pplv   c   this pplv is paired with the pslv 2 command above
  pslv 3
c part number 2
   block ...
    ...
   endpart
  pplv   c   this pplv is paired with the pslv 3 command above
 pplv   c this pplv is paired with the pslv 1 command above
 pslv 4
c part number 3
  block ...
   ...
   grep 0 1 2 3 4 5 6;
   lrep 0 1 2 3 4 5;
  endpart
 pplv   c   this pplv is paired with the pslv 1 command above
```

In this example, the first part has two nested levels of part replications from levels numbered 1 and 2. The second part has two nested levels of part replications from levels numbered 1 and 3. The third part has three levels of part replications, one each from level number 4 and global and local coordinate transformations.

pplv end replicating multiple parts

pplv *(no arguments)*

Remarks

See the remarks on **pslv**. **Pplv** marks the end of the scope of part replications begun by a **pslv** command.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

csca scale all coordinates of all following parts

csca *scale*

Remarks

This command scales all parts that follow it.

xsca scale all x-coordinates of all following parts

xsca *scale*

Remarks

This command scales the x-coordinates of all parts that follow it.

ysca scale all y-coordinates of all following parts

ysca *scale*

Remarks

This command scales the y-coordinates of all parts that follow it.

zsca scale all z-coordinates of all following parts

zsca *scale*

Remarks

This command scales the z-coordinates of all parts that follow it.

xoff translate all x-coordinates of all following parts

xoff *offset*

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Remarks

This command translates the x-coordinates of all parts that follow it.

yoff translate all y-coordinates of all following parts

yoff *offset*

Remarks

This command translates the y-coordinates of all parts that follow it.

zoff translate all z-coordinates of all following parts

zoff *offset*

Remarks

This command translates the z-coordinates of all parts that follow it.

gexch permute the coordinates of all following parts

gexch *new_x_# new_y_#*

where

new_x_#	identifies the old coordinate which will become x: 1 for x, 2 for y, 3 for z
new_y_#	identifies the old coordinate which will become y: 1 for x, 2 for y, 3 for z

Remarks

See the remarks on **exch** below. **Gexch** is the same as **exch**. If both the **exch** and **gexch** are issued, **exch** is performed first.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

exch permute the coordinates of all following parts

exch *new_x_# new_y_#*

where

new_x_# identifies the old coordinate which will become x: 1 for x, 2 for y, 3 for z

new_y_# identifies the old coordinate which will become y: 1 for x, 2 for y, 3 for z

Remarks

This command permutes the x, y, and z coordinates. It applies to all parts defined after this command is issued.

A permutation of (1,2,3) is normally denoted by three numbers, e.g. (3,2,1). There are only two arguments to this command with the third number implied.

There is no way to undo this command except by issuing another **exch** command that applies the inverse permutation.

This permutation is applied before any permutations from **gexch**.

Examples

```
exch 2 1
```

For every part after this command is issued, the x and y coordinates will be interchanged.

```
exch 1 3
```

For every part after this command is issued, the y and z coordinates will be interchanged.

nerl rule used to number the nodes and brick elements

nerl *flag*

where the *flag* can be

0	for default numbering method
1	for ordered by increasing i-index first
-1	for ordered by decreasing i-index first
2	for ordered by increasing j-index first
-2	for ordered by decreasing j-index first

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

3 for ordered by increasing k-index first
-3 for ordered by decreasing k-index first

Remarks

The default numbering is complicated to describe and is good for most problems. Nodes and brick elements can be numbered by specifying the last index so that the part can be split into layers. This way, the first sequence of nodes and elements will be found in the first layer, the second sequence of nodes and elements will be found in the second layer, and so on.

gmi material number increment for global replication

gmi *material_#_increment*

Remarks

This command sets the increment used to change the material numbers in a part that is being replicated using the **grep** command. For each replication of a part, the material numbers are increased from the previous replication. For example, if materials 1 and 6 are used in the creation of a part and if the global material increment is 3, then a replication of this part using **grep** would be assigned materials 4 and 9, respectively.

The default is 0.

Example

```
gmi 5 c set the material number increment for grep replications
lmi 1 c set the material number increment for lrep replications
block 1 2;1 2;1 2;1 2;1 2;1 2; c single element part
mate 1 c set the material number for the original part
lct 4 mx 1;repe 4; lrep 0:4;  c 5 local replications
gct 4 my 1;repe 4; grep 0:4;  c 5 global replications
endpart
```

In this example, 25 replications of a single element part are created with each element having a different material number. The material numbers are incremented according to the order of the transformations in the **lrep** and **grep** commands.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

lmi material number increment for local replication

lmi *material_#_increment*

Remarks

This command sets the increment used to change the material numbers in a part that is being replicated using the **lrep** command. For each replication of a part, the material numbers are increased from the previous replication. For example, if materials 1 and 6 are used in the creation of a part and if the local material increment is 1, then a replication of this part using **lrep** would be assigned materials 2 and 7, respectively.

The default is 0. See the example for the **gmi** command.

gsii sliding interface number increment for global replication

gsii *sliding_increment*

Remarks

This command sets the increment used to change the sliding interface numbers in a part that is being replicated using the **grep** command. For each replication of a part, the sliding interface numbers are increased from the previous replication. For example, if sliding interface numbers 1 and 6 are used in the creation of a part and if the global sliding interface increment is 3, then a replication of this part using **grep** would be assigned materials 4 and 9, respectively.

The default is 0.

Example

```
gsii 5
lsii 1
sid  1 sv; sid  2 sv; sid  3 sv; sid  4 sv; sid  5 sv;
sid  6 sv; sid  7 sv; sid  8 sv; sid  9 sv; sid 10 sv;
sid 11 sv; sid 12 sv; sid 13 sv; sid 14 sv; sid 15 sv;
sid 16 sv; sid 17 sv; sid 18 sv; sid 19 sv; sid 20 sv;
sid 21 sv; sid 22 sv; sid 23 sv; sid 24 sv; sid 25 sv;
block 1 2;1 2;1 2;1 2;1 2;1 2;
si 1 1 1 2 2 1 1 m
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
lct 4 mx 1;repe 4; lrep 0:4;
gct 4 my 1;repe 4; grep 0:4;
endpart
```

In this example, there are 25 replications. The first, with no transformation applied, has a sliding interface numbered 1. Every subsequent replication has a sliding interface that is one greater.

lsii sliding interface number increment for local replication

lsii *slide_#_increment*

Remarks

This command sets the increment used to change the sliding interface numbers in a part that is being replicated using the **lrep** command. For each replication of a part, the sliding interface numbers are increased from the previous replication. For example, if sliding interface numbers 1 and 6 are used in the creation of a part and if the local sliding interface increment is 1, then a replication of this part using **lrep** would be assigned materials 2 and 7, respectively.

The default is 0. See the **gsii** command for an example.

11. Control Statements

These statements are much like the control statements in a programming language. They are useful when creating template command files to automate mesh generation in **True*Grid*®**. A set of parameters are assigned values prior to entering these commands. The control statements make it possible to execute a different set of commands in **True*Grid*®**, depending on the parameter values.

if begin an if... elseif ... else ... endif

if (*expression* **) then**

where *expression* can be relational or logical with the following features:

Integer, floating point, and exponential numbers and parameters can be used as operands and arguments to functions.

Unitary and binary arithmetic operators + and -.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Binary arithmetic operators *, /, **, and ^.

Binary relational operators:
.gt. and > for greater than,
.ge. and >= for greater than or equal,
.lt. and < for less than,
.le. and <= for less than or equal,
.eq. and == for equal,
.ne. and != for not equal.

Binary logical operators:
.and. and & and && for the logical *and* operator,
.or. for the logical *or* operator.
.eqv. for the logical *equivalent* operator,
.neqv. for the logical *not equivalent* (exclusive or) operator.

Unitary logical operators .not. and ! .

Parenthesis to specify order of operations, where the default
order is given to exponentiation ** and ^ first,
then multiplication * and division / ,
addition + and subtraction - ,
relational operators, logical operators, and negation.

All calculations are done in floating point.
All illegal operations cause warnings.
All trigonometric operations are in degrees.
INT(x): truncates x to an integer.
NINT(x): rounds x to the nearest integer.
ABS(x): absolute value of x.
MOD(a,b): a modulo b.
SIGN(a,b): transfer the sign of b to a.
MAX(x1,x2,...,xn): maximum value of a list of numbers.
MIN(x1,x2,...,xn): minimum value of a list of numbers.
SQRT(x): square root of x where x most be positive.
EXP(x): exponential of x where x must not exceed 85.19.
LOG(x): natural logarithm where x must be positive.
LOG10(x): common logarithm base 10 where x must be positive.
SIN(x): trigonometric sine of x.
COS(x): trigonometric cosine of x.
TAN(x): trigonometric tangent of x where abs(x) can not be 90.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

ASIN(x): trigonometric arcsine of x where abs(x) can not exceed 1.
ACOS(x): trigonometric arccosine of x where abs(x) can not exceed 1.
ATAN(x): trigonometric arctangent.
ATAN2(y,x): trigonometric arctangent with two arguments.
SINH(x): hyperbolic sine.
COSH(x): hyperbolic cosine.
RAND: uniform random number of unit length with default *mea*n 0 w/ forms
RAND,
RAND(seed),
RAND(seed, mean)
where *seed* is the random number seed, and
mean is the mean of the distribution.

NORM: normal random number w/ default standard deviation 1 & mean 0, w/ forms
NORM,
NORM(seed),
NORM(seed, mean),
NORM(seed, mean, sig)
where *seed* is the random number seed,
mean is the mean of the distribution, and
sig is the standard deviation from the mean and must be positive.

Remarks

The **if** statement must be on a line by itself and it cannot be wrapped around to another line. This is like the block **if** statement in FORTRAN. Every **if** statement must be ended with an **endif**. If there is another **if** statement that follows before the **endif** statement, then the second **if** statement must be ended with an **endif** before the first one can be ended. This is called nesting of **if** statements. 19 nestings of **if** statements are allowed. Optionally, multiple **elseif** statements can be used within the **if** statement. An **else** statement can optionally be used as the final option in the **if** statement. The **elseif** and **else** can also have up to 19 nested **if** statements following it. The relational or logical expression found in the **if** statement is evaluated. This expression can have any of the features found in expressions. Relational and logical operators are also available. The order of precedence matches that of FORTRAN. If the expression is true, then all statements and commands found between the **if** and any associated **elseif**, **else**, or **endif** statement will be processed. This set of statements is referred to as the scope of the **if** statement. If the expression is false, these same statements and commands, those within the scope of the **if** statement, will be ignored. The **if** statement is a command and it cannot be used to select one set of arguments from another by embedding the **if** statement within another command.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

When two numbers, which may be from the evaluations of expressions, are compared, possible roundoff errors are estimated and are used to determine equality.

This command is typically added to a session file and rerun. It is more difficult to plan a set of commands such that you can enter the **if-elseif-else-endif** statement interactively.

If you use both the **while** and **if** statements, their scopes must obey the containment rule. Either the scope of the **while** statement must be contained in the scope of the **if** statement, the scope of the **if** statement must be contained in the scope of the **while** statement, or the two scopes are disjoint.

Example

```
if(%rebar.eq.1)then
 if(%k.ge.2*%k)then
  insprt 1 5 1 [%k]
  parameter k7 [%k1+1] k2 3;
  if(%sk.eq.1)then
   sd [%ns+1] plan %xx 0 [%zz+%k*%ze] [-sin(%p)] 0 [cos(%p)]
  elseif(%sk.le.0)then
   echo Error in parameter sk
   end
  else
   sd [%ns+1] plan %xx 0 [%zz+%k*%ze] 0 1 0
  endif
  sfi ;;-2;sd [%ns+1]
 else
  parameter k 2;
 endif
endif
```

elseif add an option to an if statement

elseif (*expression* **) then**

where *expression* is defined in the **if** statement above.

Remarks

The **elseif** statement must be on a line by itself and it cannot be wrapped around to another line. This is like the block **elseif** statement in FORTRAN. Every **elseif** belongs to an **if** statement which must be ended with an **endif.** If there is another **if** statement that follows before the **endif** statement, then

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

that **if** statement must be ended with an **endif** before the first **if**..**elseif** one can be ended. This is called nesting of **if** statements. **True*Grid*®** allows for 19 nestings of **if** statements. Optionally, multiple **elseif** statements can be used within the **if** statement. An **else** statement can optionally be used as the final option in the **if** statement. The **elseif** and **else** can also have up to 19 nested **if** statements following it. The relational or logical expression found in the **if** statement is evaluated. This expression can have any of the features found in expressions. Relational and logical operators are also available. The order of precedence matches that of FORTRAN. If the preceding associated **if** statement expression is false and all preceding associated **elseif** statement expressions are also false, then the expression in this **elseif** statement is evaluated. If it is true, then all statements and commands found between the **elseif** and any subsequent associated **elseif**, **else**, or **endif** statement will be processed. The statements and commands found between this **elseif** statement and the next associated **elseif**, **else**, or **endif** statement are referred to as the scope of the **elseif** statement. If the expression is false, these same statements and commands, the commands in the scope of this **elseif** statement, will be ignored.

When two numbers, which may be from the evaluations of expressions, are compared, possible roundoff errors are estimated and are used to determine equality.

This command is typically added to a session file and rerun. It is more difficult to plan a set of commands such that you can enter the **if-elseif-else-endif** statement interactively.

If you use both the **while** and **elseif** statements, their scopes must obey the containment rule. Either the scope of the **while** statement must be contained in the scope of the **elseif** statement, the scope of the **elseif** statement must be contained in the scope of the **while** statement, or the two scopes are disjoint.

else final option in an if statement

else

Remarks

The **else** statement must be on a line by itself. This is like the block **else** statement in FORTRAN. Every **else** belongs to an **if** statement which must be ended with an **endif.** If there is another **if** statement that follows before the **endif** statement, then that **if** statement must be ended with an **endif** before the first **if**. This is called nesting of **if** statements. **True*Grid*®** allows for 19 nestings of **if** statements. An **else** statement can optionally be used as the final option in the **if** statement. The **else** can also have up to 19 nested **if** statements following it.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

This command is typically added to a session file and rerun. It is more difficult to plan a set of commands such that you can enter the **if-elseif-else-endif** statement interactively.

If you use both the **while** and **else** statements, their scopes must obey the containment rule. Either the scope of the **while** statement must be contained in the scope of the **else** statement, the scope of the **else** statement must be contained in the scope of the **while** statement, or the two scopes are disjoint.

endif end an if statement

endif

Remarks

The **endif** statement must be on a line by itself. This is like the block **if**..**elseif**..**else**..**endif** statement in FORTRAN. Every **endif** ends an **if** statement.

endwhile end a while statement

endwhile

Remarks

The **endwhile** statement must be on a line by itself. Every **endwhile** ends or closes a **while** statement.

while begin a loop iteration

while (*expression* **)**

where *expression* can be relational or logical with the following features:

Integer, floating point, and exponential numbers and parameters can be used as operands and arguments to functions.

Unitary and binary arithmetic operators + and -.

Binary arithmetic operators *, /, **, and ^.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Binary relational operators:
.gt. and > for greater than,
.ge. and >= for greater than or equal,
.lt. and < for less than,
.le. and <= for less than or equal,
.eq. and == for equal,
.ne. and != for not equal.

Binary logical operators:
.and. and & and && for the logical *and* operator,
.or. for the logical *or* operator.
.eqv. for the logical *equivalent* operator,
.neqv. for the logical *not equivalent* (exclusive or) operator.

Unitary logical operators .not. and ! .

Parenthesis to specify order of operations, where the default
order is given to exponentiation ** and ^ first,
then multiplication * and division / ,
addition + and subtraction - ,
relational operators, logical operators, and negation.

All calculations are done in floating point.
All illegal operations cause warnings.
All trigonometric operations are in degrees.
INT(x): truncates x to an integer.
NINT(x): rounds x to the nearest integer.
ABS(x): absolute value of x.
MOD(a,b): a modulo b.
SIGN(a,b): transfer the sign of b to a.
MAX(x1,x2,...,xn): maximum value of a list of numbers.
MIN(x1,x2,...,xn): minimum value of a list of numbers.
SQRT(x): square root of x where x most be positive.
EXP(x): exponential of x where x must not exceed 85.19.
LOG(x): natural logarithm where x must be positive.
LOG10(x): common logarithm base 10 where x must be positive.
SIN(x): trigonometric sine of x.
COS(x): trigonometric cosine of x.
TAN(x): trigonometric tangent of x where abs(x) can not be 90.
ASIN(x): trigonometric arcsine of x where abs(x) can not exceed 1.
ACOS(x): trigonometric arccosine of x where abs(x) can not exceed 1.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

ATAN(x): trigonometric arctangent.
ATAN2(y,x): trigonometric arctangent with two arguments.
SINH(x): hyperbolic sine.
COSH(x): hyperbolic cosine.
RAND: uniform random number of unit length and default *mean* 0 /w forms
RAND,
RAND(seed),
RAND(seed,mean)
where *seed* is the random number seed, and
mean is the mean of the distribution.

NORM: normal random number w/ default standard deviation 1 & mean 0, w/ forms
NORM,
NORM(seed),
NORM(seed,mean),
NORM(seed,mean,sig)
where *seed* is the random number seed,
mean is the mean of the distribution, and
sig is the standard deviation from the mean and must be positive.

Remarks

The **while** statement must be on a line by itself and it cannot be wrapped around to another line. Every **while** statement must be ended with a closing **endwhile** statement. The statements and commands that fall between the **while** and closing **endwhile** statements is referred to as the scope of the command. If there is another **while** statement that follows before the closing **endwhile** statement, then the second **while** statement must be ended with an **endwhile** before the first one can be ended. This is called nesting of **while** statements. 19 nestings of **while** statements are allowed. The relational or logical expression found in the **while** statement is evaluated. This expression can have any of the features found in expressions. Relational and logical operators are also available. The order of precedence matches that of FORTRAN. If the expression is true, then all statements and commands found between the **while** and **endwhile** statement will be processed and, upon completion, this whole process is repeated until the expression is false. When the expression becomes false, these same statements and commands will be ignored and the next statement or command to be executed will be the one that follows the closing **endwhile** statement. Use the **break** feature to jump out of a while statement. The **while** statement is a command and it cannot be embedded within another command.

When two numbers, which may be from the evaluations of expressions, are compared, possible roundoff errors are estimated and are used to determine equality.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

This command is typically added to a session file and rerun. It is more difficult to plan a set of commands such that you can enter the while statement interactively.

If you use both the **while** and one of the **if-elseif-else** statements, their scopes must obey the containment rule. Either the scope of the **while** statement must be contained in the scope of the **if-elseif-else** statement, the scope of the **if-elseif-else** statement must be contained in the scope of the **while** statement, or the two scopes are disjoint.

Example

```
para n 0;
while(%n.lt.10)
 block 1 2;1 2;1 2;0 1 0 1 0 1
 tri ;;; v %n %n %n;
 endpart
 para n [%n+1];
endwhile
```

This example produces 10 unit brick elements along a diagonal.

12. Merging Nodes

In the Merge phase, nodes that are close to one another are merged into a single node. Tolerance commands allow you to define how close is close. All tolerances are in absolute distances. There are commands for specifying tolerances for the general merging of all nodes over all parts or just nodes on the exterior faces of the mesh. There are commands for specifying the tolerances for the special merging of nodes between parts or within a part. These special tolerances override the general ones. If no tolerance commands are specified, then no merging is done. However, the Merge phase must be entered in order to build the node map which is used to generate output for a simulation code.

Invocation of a tolerance command (**t**, **tp**, **st**, **stp**) within the Merge phase causes an immediate merging of nodes. These commands can also be invoked any time; when the Merge phase is entered, those tolerance commands are immediately executed or re-executed as the case may be. The merge process is always performed on the nodes in their original (prior to any merging) state. Merging is not cumulative. If you leave the Merge phase and reenter it, all merging is recalculated with what ever new parts that have been added. This lets you interactively experiment with merging and tolerances. Setting a tolerance to a negative value is an easy way to restore the nodes to their original states. Graphical displays of the mesh in the Merge phase always reflect the results of any merging.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Nodes are merged depending on the distance between them. If a node lies within a tolerance distance of more than one other node, then it is merged with the closest one. When merging several nodes into one node, the first-defined node survives. This can be overridden by the **bptol** command. Nodes within a joint, spring, or weld spot and across the two sides of a sliding surface are not merged. When the first merging of nodes occurs, a sliding interface table is calculated which is used in the merging process. This table is written to the screen and to the save file and is intended as diagnostics. The following is a sample of that table:

```
                SLIDING INTERFACE SUMMARY
Surf    S-node    S-lseg    S-qseg    M-node    M-lseg    M-qseg
   1       105        84         0       468       418         0
   2       232         0        52       468       418         0
   3       221         0         0       390       304         0
   4       221         0         0       390       304         0
   5       158         0         0       120        88         0
   6       158        30        30       120        88         0
   7       204       102         0       204       102         0
   8       232         0        52        90        52         0
   9       101        18        18       161       132         0
  10       101        18        18       161       132         0
  11       548       120       120      3216         0      1056
  12       133        84         0       161       132         0
  13       133        84         0       161       132         0
  14       308       240         0      3216         0      1056
```

This table is organized by the sliding interface number on the right. Columns 2, 3, and 4 are datum pertaining to the slave side on the interface; columns 5, 6, and 7 to the master side. Columns 2 and 5 (`S-node` and `M-node`) are node counts. Columns 3 and 6 (`S-lseg` and `M-lseg`) are linear face counts, and columns 4 and 7 (`S-qseg` and `M-qseg`) are quadratic face counts.

A table of merged nodes is always written after the **tp** or **stp** commands are executed.

```
                MERGED NODES SUMMARY
  12 nodes merged between parts        1 and        2
  16 nodes merged between parts        2 and        2
  12 nodes merged between parts        1 and        3
  16 nodes merged between parts        3 and        3
 216 nodes merged between parts        4 and        4
  30 nodes merged between parts        7 and        7
  88 nodes merged between parts        8 and        8
 390 nodes were deleted by tolerancing
```

Up through 4000 parts can be merged under general tolerancing (i.e. no use of the **ptol** or **bptol** commands). 300 parts can be merged under special tolerancing (**ptol** and **bptol**).

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

The following is a common error to avoid. Suppose you create three parts that meet as shown in the figures below. Then define a sliding interface between parts 1 and 2 and also between parts 1 and 3. No nodes will be merged between parts 1 and 2 and between parts 1 and 3. However, nodes can be merged between parts 2 and three. Sometimes you need to look closely in the graphics or carefully check the Merged Nodes Summary to detect this error. To fix this error, if indeed it is an error, use a dummy sliding interface between parts 2 and 3 to force no merging between those parts. Alternatively, use the **bptol** command with a negative number to avoid merging between those parts. You should also consider extending both interfaces 1 and 2 across to parts 3 and 2, respectively, because they may come in contact. This is an ambiguous situation since there are equally plausible situations where parts 2 and 3 should be merged together.

```
sid 1 sv;
sid 2 sv;
block 1 3;1 3;1 3;
     1 2 1 2 1 2
sii -2;;;1 s;
sii ;-2;;2 s;
block 1 3;1 3;1 3;
     2.1 3 1 2 1 2
sii -1;;;1 m;
block 1 3;1 3;1 3;
     1 2 2.1 3 1 2
sii ;-1;;2 m;
merge
stp .2
```

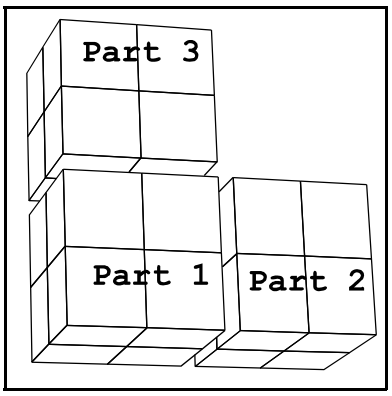


Figure 285 Before **stp**

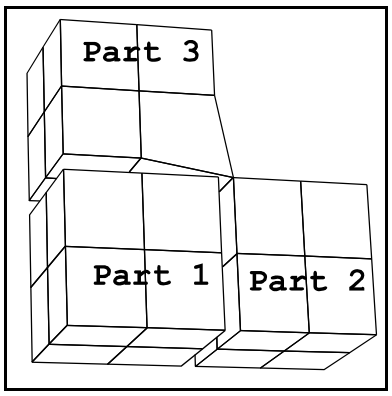


Figure 286 After **stp**

bnstol between node set tolerance

bnstol *node_set*$_1$ *node_set*$_2$ *tolerance*

where

node_set$_1$	first set of nodes to be compared
node_set$_2$	second set of nodes to be compared
tolerance	tolerance that must be met for merging between node

Remarks

A pair of nodes may be merged into one node based on this new command. If one node of the pair is in one of these two node sets and the other node is in the other node set, they will be merged into one node if the distance between them is less than the *tolerance*. This is checked for all possible pairs

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

of nodes. This command works similarly to the **ptol** and **bptol** commands. These commands only set the tolerance to be used when a merge command, such as **stp** or **tp**, is issued.

It is possible with the **ptol**, **bptol**, and the **bnstol** commands for a pair of nodes to have several tolerances, making the merging process ambiguous. To resolve this issue, when the merging of a pair of nodes are ruled by the issuing of several of these commands, then the one that was issued last will be the command to determine if the pair of nodes are to be merged. The key to this is that the order that these commands are issued can be important.

For example, one may wish to merge nodes in a specific region and rule out merging in all other regions. This can be done by creating two node sets, each set containing the nodes on opposite faces to be merged and issuing the **bnstol** command. Then an **stp** command with a tolerance of -1 could be issued to rule out all merging except for the region of the two node sets.

This feature has the opposite effect of the sliding interface (**sid** command) of type dummy.

merge switch to the merge (assembly) phase

merge *(no arguments)*

Remarks

This command does not work in the merge phase since you will already be in the merge phase. If this command is issued while in the part phase, it will cause the part to end. This command is useful in the control phase to switch to the merge phase where there is a 3D graphics window.

mns merge node sets

mns *mns# node_set_1 node_set_2*

where

mns#	mns identification number
node_set_1	name of the first node set
node_set_2	name of the second node set

Remarks

Two node sets are specified and any pair of nodes, one in each set, are allowed to be merged if they are found on opposite sides of a sliding interface. A merge command in the Merge phase is still

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

required for the nodes to be merged, and only those nodes within the specified distance between them will be merged. Up to 15 pairs of node sets are allowed.

This feature can be undone by using a ";" for one of the sets in order to prematurely end the command.

Example

Two shell parts are generated with a small gap between them. The first part is assigned to be the master side of a sliding interface, with the faces oriented towards the second part. The second part plays the opposite role as the slave side, facing the first side. Two node sets are created, one from the first part along an edge, and the second node set along the corresponding edge of the second part. These nodes along the corresponding edges are allowed to merge.

```
sid 1 sv;
block 1 11;1 11;-1;-1 1 -1 1 0
orpt + 0 0 1
si 1 1 1 2 2 1 1 m;
nset 1 1 1 1 2 1 = ifc1
block 1 11;1 11;-1;-1 1 -1 1 .001
orpt + 0 0 -1
si 1 1 1 2 2 1 1 s;
nset 1 1 1 1 2 1 = ifc2
merge
mns 1 ifc1 ifc2
stp .0011
```

st set tolerance and merge surface nodes

st *tolerance*

Remarks

Only nodes that lie on the surfaces or exterior faces of parts are considered for merging. Exterior faces include faces of the mesh that are physically matching but are logically distinct faces of the mesh. No table of results is printed. If the sliding interface table has not yet been printed, it will be printed at this time.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

stp set tolerance and merge surface nodes, with diagnostics

stp *tolerance*

Remarks

This command performs as **st** except that the results of the merge are reported. Exterior faces include faces of the mesh that are physically matching but are logically distinct faces of the mesh. A table of the merged nodes is printed to the screen and to the save file. If there are any sliding interfaces, then the first merge command will cause a sliding interface report to be written first.

Example

In the example above for the **mns** command, the **stp** command would cause the following tables to be written.

```
stp .002
                SLIDING INTERFACE SUMMARY
   Surf   S-node   S-lseg   S-qseg   M-node   M-lseg   M-qseg
      1      121      100        0      121      100        0
                  MERGED NODES SUMMARY
         11 nodes merged between parts        1 and        2
         11 nodes were deleted by tolerancing
```

t set tolerance and merge nodes

t *tolerance*

Remarks

All nodes of all parts are considered for merging. If the sliding interface table has not yet been printed, it will be printed at this time.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

tp set tolerance and merge nodes, with diagnostics

tp *tolerance*

Remarks

This command performs as **t** except that the results of the merge are reported. See the **stp** command for an example. If the sliding interface table has not yet been printed, it will be printed at this time.

ztol minimum non-zero absolute coordinate

ztol *tolerance*

Remarks

This command causes each coordinate of each node, whose absolute value is less than *tolerance*, to be set to zero prior to merging and prior to the generation of output. Each invocation of **ztol** causes the **original** coordinates to be compared against *tolerance*. This lets you interactively experiment with **ztol**. This command does not change the data base. It only affects the merging process and the output.

When a node is picked in the merge phase, the environment window displays the coordinates. **Ztol** affects these coordinates displayed in the pick panel.

bptol between parts tolerance specification

bptol $part_1$ $part_2$ *tolerance*

where

$part_1$ and $part_2$	are part numbers, and
tolerance	is the between part tolerance.

Remarks

The *tolerance* is used as the between-part tolerance for merging nodes between parts $part_1$ and $part_2$. This command does not initiate the merge procedure. The *tolerance* overrides the default tolerance specified by the **t**, **tp**, **st**, and **stp** tolerance commands just between these two parts. When nodes are merged, the nodes belonging to part $part_1$ survive. This is also a way to change the ordering of the

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

merging procedure. Use a negative number to avoid merging between two parts. There is a maximum of 1000 **bptol** and **ptol** commands.

This command can be easily abused. Most meshes do not need this command to force the merging. If the nodes between parts do not merge with a reasonable tolerance using, for example, the **stp** command, then perhaps the interface between the two parts were not generated identically. This may warrant some investigation into the cause. Many of these problems can be avoided by using the **bb** command.

ptol part tolerance specification

ptol *part_number tolerance*

Remarks

The *tolerance* is used as the absolute tolerance for merging nodes within part *part_number*. This command does not initiate the merge procedure. The *tolerance* overrides the default tolerance specified by the **t**, **tp**, **st**, and **stp** tolerance commands just within this part. Use a negative number to avoid merging within the part. There is a maximum of 1000 **bptol** and **ptol** commands.
This command can be easily abused. Most meshes do not need this command to force the merging. If the nodes within a part do not merge with a reasonable tolerance using, for example, the **stp** command, then perhaps the interfaces within the part were not generated identically. This may warrant some investigation into the cause.

rigbm identify two rigid bodies to be merged

rigbm $material_1$ $material_2$

where

$material_1$ $material_2$ must be rigid materials in DYNA3D or LS-DYNA.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

13. Global Graphics Commands

The following commands are available at any time.

ibzone control the computational window frame

ibzone *option*

where the *option* can be

on *size*	turn on the frame and set the size of the index zone
off	turn off the frame

Remarks

This command activates or deactivates the drawing of the boundary between part and index-bar on the computational window. The outer part of the window is the zone of influence of the index-bar. This is where the mouse can move to cause highlighting of the mesh in both the physical and computational window. This is also the area where a mouse click or a click-and-drag can make selections in the mesh. The inner box of the zone becomes the area where the computational window can be seen and where a click-and-drag can select a region directly from the picture, instead of from the index bar. One can also change the size of the zone. The default is **on 1.0**.

noplot turn all graphics off

noplot *(no arguments)*

Remarks

This command is useful when you have a batch file with some graphics commands. When running this batch file, you may wish to suppress the graphics. It will speed the process. If you wish to avoid all graphics including the creation of the graphics windows, use the **nogui** option on the execute line. If you use the **nogui** option, the **noplot** command is not needed to suppress the graphics.

plot turn graphics back on

plot *(no arguments)*

Remarks

This command re-activates the graphics after issuing the **noplot** command. Graphics are active by default.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

14. Miscellaneous Commands

becho echo and beep

becho *text*

Remarks

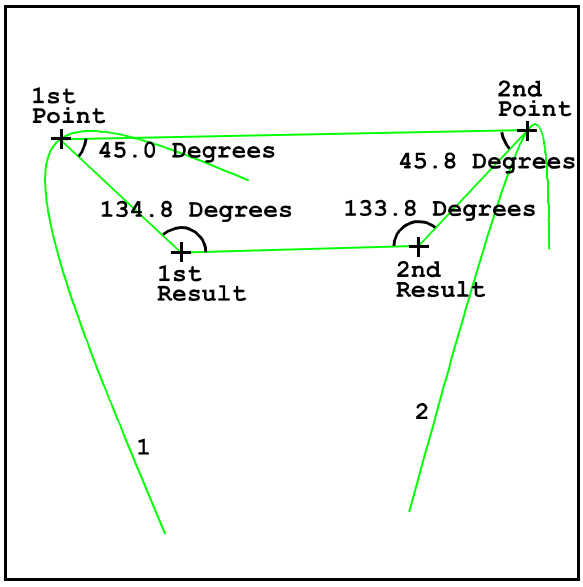
A string of up to 80 characters is printed in the text window and the computer beeps.

bulc locate butterfly triple point

bulc *curve*$_1$ *curve*$_2$ *x y z ratio*

Remarks

This command locates an interior vertex of a butterfly structure based only on the shape of two exterior curves. The goal is to produce a pair of points from the two exterior curves and two additional points from this command to form a trapezoid with angles approximately 45, 45, 135, and 135 degrees. This command is needed twice to form the two interior points. The coordinates are saved in the parameters %xprj, %yprj, and %zprj. The point that is input is used to select the closest point on each of the two specified curves. Then a right isosceles triangle is fitted through these points in a plane that is nearly orthogonal to both curves. Actually, there are two such triangles that are formed. The **orpt** command is used to select one from this pair. Then the leg of the triangle that is closest to the initial point is chosen. A point is selected along this leg of the triangle so that the ratio of the leg of the resulting trapezoid and the base of the trapezoid matches the specified ratio.



Trapezoid using bulc

Example

```
curd 1 csp3 00
   -4.7833413e-01   2.6501161e-01
-8.3939201e-01
   -7.0889658e-01   6.8763685e-01
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
-1.3136336e-01
 -5.9971255e-01 4.8465800e-01 6.3912874e-01;;;
curd 2 csp3 00
 -6.2983650e-01 -1.9638570e-02 -7.7306795e-01
 -9.9466640e-01 9.3395844e-02 -3.7190955e-02
 -6.6715145e-01 4.7235709e-02 7.4401599e-01;;;
orpt + 0 0 0
bulc 1 2 -9.9454612e-01 9.5007576e-02 5.5500008e-03 .5
c   -7.752336E-01   1.707245E-01   9.306287E-03
bulc 1 2 -7.1137702e-01 6.9195741e-01 -3.5788484e-02 .5
c   -6.328765E-01   4.716224E-01  -3.093115E-02
```

c beginning of a comment

c *text*

Remarks

The character **c** represents a comment when it appears at the beginning of a line and is followed by a space, or when it appears in the middle of a line and is both preceded and followed by a space. Comments will be preserve in the tsave (session) file.

This feature is especially useful for commenting out commands in a batch input file. Another way to insert comments is to use a dollar sign. If you have a large body of text or a section of commands you wish to turn into comments, encase the text with the curved brackets {...}. This works across multiple lines of text.

circent center of a circle

circent $x_1\ y_1\ z_1\ x_2\ y_2\ z_2\ x_3\ y_3\ z_3$
where (x_i, y_i, z_i) is a point on the circle for i=1,2,3

Remarks

This command finds the center of a circle that passes through 3 non-planar points.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
circent 1 1 1 2 -2 2 -3 -3 3
 center =  -8.71622E-01  -1.11486E+00   2.02703E+00
 radius =    3.0050633
 normal =  -1.16248E-01  -3.48743E-01  -9.29981E-01
```

crprod cross product

Calculates the cross product of two vectors.

crprod $x_1\ y_1\ z_1\ x_2\ y_2\ z_2$

Example

```
crprod [cos(70)] [sin(70)] 0 [cos(160)] [sin(160)] 0
c  (x,y,z)=   0.00000E+00   0.00000E+00   1.00000E+00
c  normalized (x,y,z)=   0.00000E+00   0.00000E+00   1.00000E+00
```

curtyp default attach command

curtyp *type*

where *type* can be

cur	equivalence the attach button to the **cur** command
curs	equivalence the attach button to the **curs** command
cure	equivalence the attach button to the **cure** command
curf	equivalence the attach button to the **curf** command

Remarks

This command controls the type of 3D curve attachment used when the ***attach*** button is selected in the environment window while in the part phase. The following summarizes the differences between these options. See the full description of each of these commands for further information.

cur: the selected edge is attached to a curve with the end vertices placed at the closest points on the curve. All interior nodes, including any interior vertices are distributed along the curved, based on the position of the end vertices. Essentially, the interior vertices have no degrees of freedom.

curs: (Default) each component edge of the selected composite edge is placed on the curve, independently. This means that each vertex along the selected component edge is placed on the curve independently. Then the interior nodes of each component edge is distributed along the curve based on its end vertices. The same thing can be accomplished by using the **cur** command for each

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

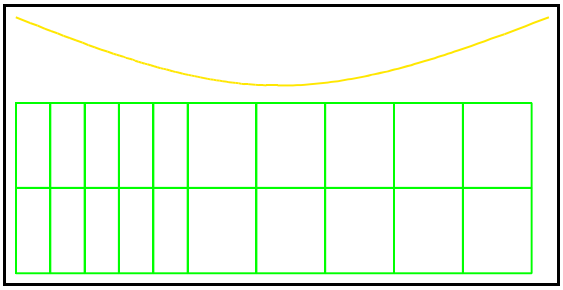
component edge of a composite edge. The **curs** command has the advantage that only one command is needed and if a partition is added with the **insprt**, the **curs** will do the interpolation that is usually expected.

cure: the end vertices of the selected edge are placed at opposite ends of the 3D curve, depending on their initial positions. Then the same interpolation as **cur** is applied. **curf**: the edge is placed onto the 3D curve just as in the **cur** command. However, the nodes are frozen to this 3D curve. Projections to surfaces will have no affect on this edge.

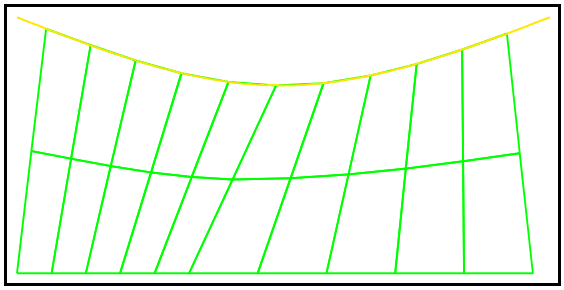
Examples

The following examples all start with the same common input. A simple 2 block part is used with one composite edge attached to a 3D curve. This demonstrates the differences between the **cur**, **curs**, and **cure** commands.

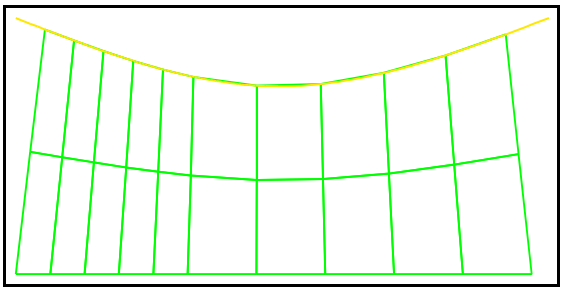
```
block 1 6 11;1 3;-1;.5 1 2 0 .5 0
curd 1 csp3 00 .5 .75 0 1.25 .55 0 2.05 .75 0 ;;;
```



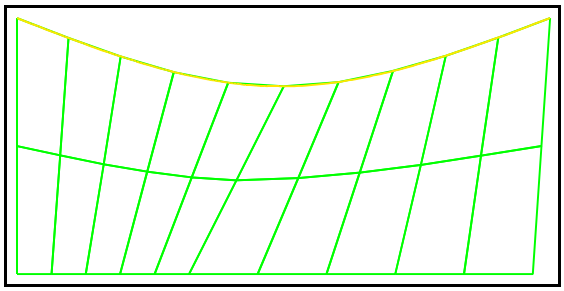
Part & Curve before attachment



Cur attachment



Curs attachment



Cure attachment

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

dc desk calculator

Prints the value of a Fortran-like expression.

dc *expression*

Remarks

With **dc**, you can see the value of an expression. For details on the syntax of the expression, see the discussion on expressions at the beginning of this section. The expression must end at the end of the line.

Example

```
para c2 2 c1 1 c0 -1;
dc (-%c1+sqrt(%c1^2-4*%c2*%c0))/(2*%c2)
 =  5.000000E-01
dc (-%c1-sqrt(%c1^2-4*%c2*%c0))/(2*%c2)
 = -1.000000E+00
```

distance distance between two points

distance $x_1\ y_1\ z_1\ x_2\ y_2\ z_2$

Remarks

This command calculates the distance between two points. Use one of the many ways to select a point in the picture using the mouse. Type the **F7** function key to print the coordinates of that point into the **distance** dialogue box or into the text window, if you are typing commands. Select the second point in the picture and type the **F7** function key again. Execute the command to get the distance.

Example

```
distance [cos(45)] [sin(45)] 0 [2*cos(45)] [2*sin(45)] 0
distance =    1.0
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

echo echo a string

echo *text*

Remarks

A string of up to 80 characters is printed in the text window.

end terminate True*Grid*® with no more output

end *(no arguments)*

errmod set error handler mode

errmod *mode*

where *mode* can be

0	for warning and error messages (default)
1	for avoiding warning messages
2	for error interrupt mode
3	for avoiding warning messages and error interrupt mode

Remarks

These options affect the way warnings and errors are handled.

If the mode 1 or 3 is selected, no warning messages will be issued.

If the mode 2 or 3 is selected, then when an error in the input is encountered while in batch mode, **True*Grid*®** will become interactive as though it had encountered an **interrupt** command in the batch file. Any problems due to the error can then be fixed interactively. Then the **resume** command can be issued to continue the batch command file. This is not easy to do sometimes. Here are three cases that one should be aware of:

1. The command in the batch file is in error and it has many arguments. Any arguments that remain after the error is detected must still be processed once the **resume** command is issued. These arguments will also cause an error and a subsequent interrupt. Warning: do not try to modify the command file at this stage. It will only cause more problems since this command file is being buffered.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

2. Some errors are not detected until the **endpart** command is encountered in the batch file. This will cause the part to be completed with errors. Then the interactive mode will be entered in the control phase. There will be little that can be done abut the part at this stage, but you will be able to proceed with a **resume** command if you wish.

3. When an **insprt** command is encountered, errors in some previously issued commands, such as the **bb** and **trbb** commands, may be detected. This is because the **insprt** command, like the **endpart** command, cause the entire mesh to be re-calculated.

expressions FORTRAN-like expressions

This FORTRAN-like expression interpreter is not a command but is a general feature that can be used anyplace in any command that requires a number. You do this by enclosing the expression in square brackets. For example, you may wish to make the number of elements within a single region of the mesh be a function of a density parameter:

```
block 1 [1+2.7*%d];1 [1+3.2*%d];1 [1+4.9*%d];0 2.7;1 4.2;-1 3.9;
```

You could calculate the distance between two points with

```
[ sqrt(%a*%a + %b*%b) ]
```

For more information on the use of parameters, see the **para** command. Such an expression can be continued across multiple lines. The closing square bracket terminates the expression. There is a limit of 240 symbols in an expression.

Expressions are also used to define a function, a function curve, a function surface, and some loads. The square brackets are not used in this case and to continue a long expression to another line, end the line with a space and ampersand, ("&").

You can enter numbers in the usual FORTRAN formats for integer, floating point, and exponential numbers.

The expressions feature can interpret the arithmetic operators + and - and the binary arithmetic operators: + - * / ** and ^ . All calculations are done in floating point. All trigonometric functions are for angles in degrees.

This feature also interprets the following FORTRAN-like functions:

int(*x*) truncates *x* to an integer

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

nint(*x***)** rounds *x* to the nearest integer
abs(*x***)** absolute value of *x*
mod(*a***,***b***)** *a* modulo *b*
sign(*a***,***b***)** transfer the sign of *b* to *a*
max(x_1,x_2,...,x_n**)** maximum number
min(x_1,x_2,...,x_n**)** minimum number
sqrt(*x***)** square root of *x*; *x* must be positive
exp(*x***)** exponential of *x*; *x* must not exceed 85.19
log(*x***)** natural logarithm of *x*; *x* must be positive
log10(*x***)** common logarithm (base 10) of *x*; *x* must be positive
sin(*x***)** sine of *x* (in degrees)
cos(*x***)** cosine of *x* (in degrees)
tan(*x***)** tangent of *x*; *x* cannot be 90, -90, 180, -180, etc.
asin(*x***)** arc sine (in degrees) of *x*; *x* must be between -1 and 1.
acos(*x***)** arc cosine (in degrees) of *x*; *x* must be between -1 and 1.
atan(*x***)** arc tangent of *x*
atan2(*y***,***x***)** arc tangent of *y*/*x*
sinh(*x***)** hyperbolic sine of *x*
cosh(*x***)** hyperbolic cosine of *x*
rand pseudo-random number from a uniform distribution of unit length, *mean* 0
rand(*seed***)** pseudo-random number from a uniform distribution of unit length, *mean* 0, computed from the given *seed*
rand(*seed*,*mean***)** pseudo-random number from a uniform distribution between *mean*-½ and *mean*+½, computed from the given *seed*
norm pseudo-random number from a normal distribution with mean 0 and standard deviation 1
norm(*seed***)** pseudo-random number from a normal distribution with mean 0 and standard deviation 1, computed from the given *seed*
norm(*seed*,*mean***)** pseudo-random number from a normal distribution with mean *mean* and standard deviation 1, computed from the given *seed*
norm(*seed*,*mean*,*sig***)** pseudo-random number from a normal distribution with mean *mean* and standard deviation *sig*, computed from the given *seed*; *sig* must be positive

For the random number functions, the default seed is 0.0.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

def define a function

def *function_name* **(***a,b,...***) =** *expression*

Remarks

A function can be used in any expression or equation. Following the function name is a pair of parenthesis enclosing a list of dummy arguments that are used in the expression which defines the function. The dummy arguments, unlike parameters, do not have a % character at the beginning of the use of a dummy argument in the definition of the function.

Example

```
def len(x1,y1,z1,x2,y2,z2)=sqrt((x1-x2)**2+(y1-y2)**2+(z1-z2)**2)
dc len(1,1,1,2,2,2)
 =  1.732051E+00
dc sqrt(3)
 =  1.732051E+00
```

include execute commands from batch file

include *filename*

Remarks

The specified file will be executed as a batch command file. The commands you have previously issued will remain in effect. Errors will be handled as usually when in batch mode.

The tsave (session) file will incorporate all of the commands that you have issued, whether interactive, from the original batch file, or from an included batch file. Thus, no **include** commands will appear in the tsave file.

The **include** statements can be nested up to 19 statements deep. There is no limit on the number of non-nested **include** commands.

If you forget to select an input file, type or use the include dialogue to initiate the execution of a command file.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

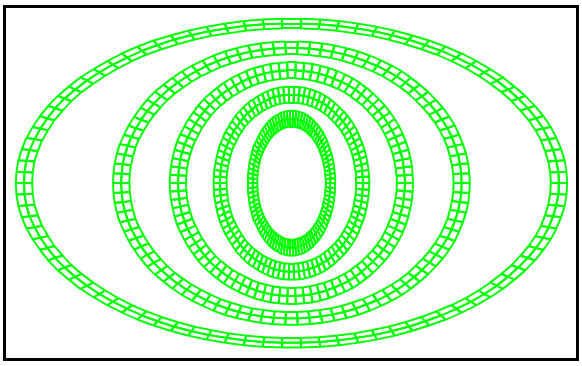
Example

Suppose there is a file named ring.inc with the following contents:

```
cylinder 1 3;1 91;-1;%r1 %r2;0 360;0;
lct 1 xsca %xs ysca %ys;lrep 1 ;
endpart
```

Then the following commands will produce the set of rings shown below:

```
para r1 .7 r2 .9 xs .6 ys 1;
include ring.inc
para r1 1 r2 1.2 xs .8 ys 1;
include ring.inc
para r1 1.3 r2 1.5 xs 1 ys 1;
include ring.inc
para r1 2 r2 2.2 xs 1 ys .8;
include ring.inc
para r1 3.2 r2 3.4 xs 1 ys .6;
include ring.inc
merge
```



Concentric rings w/ parameters & include

inprod inner or dot product

This command computes the inner or dot product of two vectors.

inprod $x_1\ y_1\ z_1\ x_2\ y_2\ z_2$

Example

```
inprod [cos(20)] [sin(20)] 0 [cos(30)] [sin(30)] 0
 dot product =   9.84808E-01
dc cos(10)
 =  9.848077E-01
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

interrupt switch from batch commands to interactive mode

When **True*Grid*®** is executing batch commands, this command halts batch execution and begins interactive execution; i.e. with keyboard and mouse.

interrupt <*no arguments*>

Remarks

Upon reaching this command in a batch file, **True*Grid*®** halts execution of commands in the file and becomes interactive. This gives you an opportunity to interactively modify the model. When you wish to continue execution of the batch file, you can issue the **resume** command.

This feature is useful in making a change to an existing mesh. Using a text editor, enter the interrupt command into the command file at the end of the list of commands for each part that needs modification. Then run the command file. When **True*Grid*®** becomes interactive issue the commands to make the desired modifications and click on the ***Resume*** button (or type the **resume** command into the test window).

Keep in mind that **True*Grid*®** will start executing commands from the command file from where it stopped after encountering the **interrupt** command. If, for example, there was an **endpart** command just after the **interrupt** command in the command file, then it would cause an error if you were to type the **endpart** command into the text window before you resumed execution of the commands from the command file. **True*Grid*®** would see two **endpart** commands in sequence, and the second one would be an error.

After you have completed the execution of all of the commands from the command file and the commands you issued interactively, you can exit **True*Grid*®** and save the new sequence of commands found in the session file (tsave file).

This procedure can also be used to hunt down a bug in your command file. After entering the interactive mode, use the ***History*** window to help locate the command(s) in error. This is also a good way to understand the topology and techniques used to create a model that you are not familiar with.
.

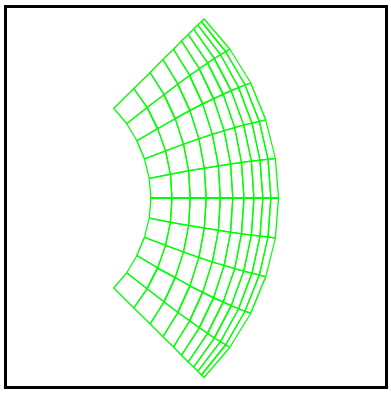
Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

intyp set default mesh interpolation

intyp *option*

where *option* can be

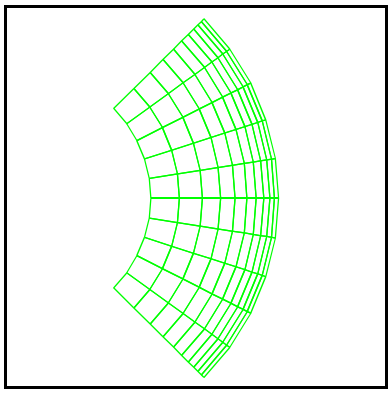
1	linear interpolation (initial default) (**lin** and **lini** commands)
2	transfinite interpolation (**tf** and **tfi** commands)



Linear Interpolation

Remarks

The linear interpolation is cheap and is good for most problems. The transfinite interpolation is better when there is a lot of curvature and when the nodes along the edges are no distributed evenly.



Transfinite Interpolation

Example

```
block 1 11;1 11;-1;1 3 -1 1 0
sfi -1;;-1; cy 0 0 0 0 0 1 1
sfi -2;;-1; cy 0 0 0 0 0 1 2
sfi 1 2;-2;-1; plan 0 0 0 1 -1 0
sfi 1 2;-1;-1; plan 0 0 0 1 1 0
res 1 1 1 2 2 1 i .8
```

painfo print information about parameters

This command gives you information on the currently defined parameters.

painfo *<no arguments>*

Remarks

This command helps you remember what you did with the **para** command.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

para define parameters

para $name_1$ $value_1$ $name_2$ $value_2$ ... $name_n$ $value_n$;
or
parameter $name_1$ $value_1$ $name_2$ $value_2$ ... $name_n$ $value_n$;

Remarks

This command is used to define parameters. Each is a symbol that stands for a number. They can be used in any other commands that need numbers or expressions. You can define up to 10,000 parameters.

Parameters enable you to set up your model in such a fashion that you can quickly and easily change the design whenever you want.

The arguments to the **para** command are pairs, first the symbolic name and then its value. The value can be a number or an expression. This value will be used until you redefine the parameter with another **para** command.

To reference a parameter, put a **%** before its symbolic name. For example:

```
para  a 1.2  b [ sqrt(%a) ]
sd  1  plan 0 0 %b 0 0 1
```

will define a plane passing through the point

$$(x = 0,\ y = 0,\ z = \sqrt{1.2}\ )$$

A parameter can be used in place of any number. It can be used in expressions and equations. It can be used to assign a value to another parameter. In all cases, you must assign a value to the parameter before referencing it. An error message will be issued if you reference a parameter without a value. All parameters are stored internally as floating point values. If a parameter is used for an integer, then the value will be truncated when it is used.

For the name of the parameter, you can use any character string that obeys the following rules:

- characters may be any of the standard (ASCII) printing characters except for:
  + , % * / ^ ( ) [ ] = & $
- the first 16 characters must be unique
- the name must be unique when case is ignored

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

To see the currently defined parameter values, issue the **painfo** command.

There are some predefined parameters. They are:

nextsrf - 1 greater than the largest surface number - see the **sd** command
nextcrv - 1 greater than the largest 3D curve number - see the **curd** command
nextlc - 1 greater than the largest load curve number - see the **lcd** command
nextln - 1 greater than the largest 2D curve number - see the **ld** command
nextmat - 1 greater than the largest material number - see the material menu
nextbb - 1 greater than the largest block boundary interface number - see the **bb** command
node - the node number from the **ajnp** command or from the node selection in the GUI
xprj - x-coordinate due to the **project** command acting on a single node
yprj - y-coordinate due to the **project** command acting on a single node
zprj - z-coordinate due to the **project** command acting on a single node
xnrm - x-component of the normal to the 1st surface from the **project** command
ynrm - y-component of the normal to the 1st surface from the **project** command
znrm - z-component of the normal to the 1st surface from the **project** command
pi - 3.1415926...
mxiridx - maximum i-index of the present part
mxjridx - maximum j-index of the present part
mxkridx - maximum k-index of the present part

resume resume executing batch commands

resume <*no arguments*>

Remarks

This command resumes execution of a batch file after the **interrupt** command has halted it. See the discussion on the **interrupt** command for the usage of the **interrupt** and **resume** commands.

title assign title to the problem

title *text*

Remarks

This command assigns a title to the model. The title should be a one-line description of the model.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

The title is used as the default graphics caption and is usually found somewhere in the output file. You can change the caption without changing the title by issuing the **caption** command.

tpara typed parameters

tpara *type symbol_1 value_1 type symbol_2 value_2 ... ;*

Remarks

This is used to pass parameters into REFLEQS, a fluids simulation code.

tricent find optimal center of a triangular structure

tricent *type* x_1 y_1 z_1 x_2 y_2 z_2 x_3 y_3 z_3

where *type* can be

1	coordinates at the corners
2	coordinates along the midpoint of edges

where

x_1 y_1 z_1	Cartesian coordinates of first point
x_2 y_2 z_2	Cartesian coordinates of second point
x_3 y_3 z_3	Cartesian coordinates of third point

Remarks

The optimal is where the three edges that meet at the center, meet at 120 degrees. There are two options. The first option bases the calculation of the coordinates of the three corners of the triangle. It assumes the midpoints of the three edges of the triangle will be connected to the center vertex. In the second option, the intermediate points along the three edges are used to calculate the optimal position for the center vertex. The latter is more versatile since it does not assume midpoints along the edges. The coordinates of the center point are immediately stored in the predefined parameters %xprj, %yprj, and %zprj.

Examples

Both of the following examples start with this input:

```
block 1 6 11;1 6 11;-1;0 1 2 0 1 2 4
sd 1 sp 0 0 0 4
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
sfi ;; -1; sd 1
dei 2 3; 2 3; -1;
pb 2 2 1 3 3 1 xy 1 1

tricent 1 0 0 4
  1.7888 0 3.5777
  0 1.7888 3.5777
pb 2 2 1 2 2 1 xyz
  %xprj %yprj %zprj
```

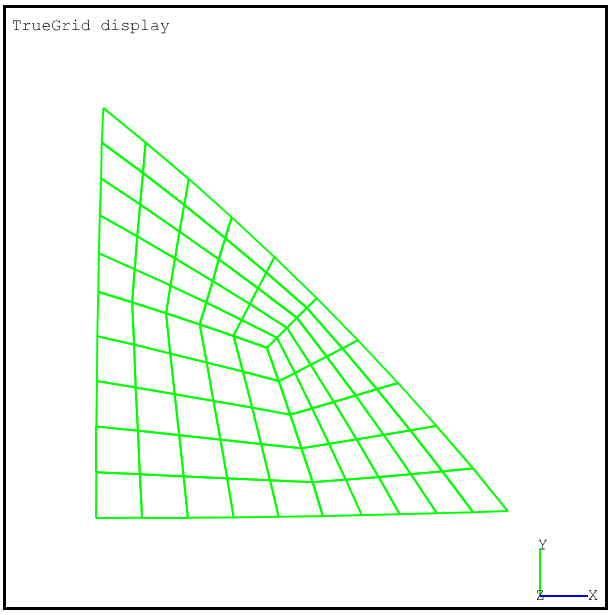


Figure 295 Optimal center based on corners

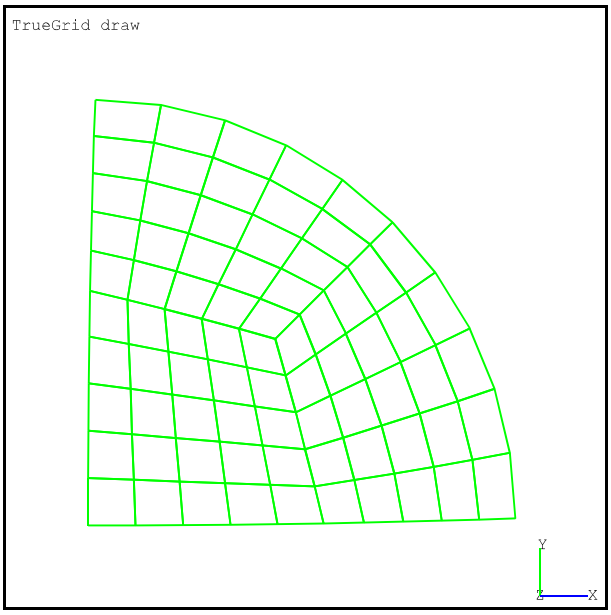


Figure 296 Optimal center based on edges

```
sd 2 cy 0 0 0 0 0 1 2
sd 2 cy 0 0 0 0 0 1 1.789
sfi 1 2; -3; -1; sd 2
sfi -3; 1 2; -1; sd 2
tricent 2 .97 0 3.88
  1.265 1.265 3.577
  0 9.7 3.88
pb 2 2 1 2 2 1 xyz
  %xprj %yprj %zprj
```

subang angle between two intersecting lines

subang $x_1\ y_1\ z_1\ x_2\ y_2\ z_2\ x_3\ y_3\ z_3$

Remarks

The angle subtended by two intersecting lines is printed.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Example

```
suban 1 1 [sqrt(2)] 0 0 0 1 1 [-sqrt(2)]
angle =    90.0
```

caption assign a caption to the physical window

caption *option*

where the *option* can be

cap *text*
on
off

Remarks

The title defined with the **title** command is the default caption.

mxp change number of mesh convergence passes

mxp *#_passes*

Remarks

This command is used to handle rare cases where inter-dependencies in the building of a part are not fully resolved. This can occur when there are many interpolations/smoothing commands that intersect and cross each other.

This is the maximum number of passes used in the projection algorithm to resolve dependencies within the **block**, **cylinder**, and **blude** parts. These dependencies occur when a multiple region is interpolated and its boundary is the interior of another multiple region interpolation. The default is 4. The minimum is 1. 2 handles almost all cases. If there are dependencies, it will take a longer time to build the part when the maximum number of passes is increased.

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

trapt transform a point

trapt *x y z type id*

where the *type* can be

lct for a local coordinate transformation

gct for a global coordinate transformation

where *id* is the number of the transformation

Remarks

Trapt will transform a 3D coordinate using a local or global coordinate transformation formed in the **lct** or **gct** commands, respectively. The results are stored in the parameters %xprj, %yprj, and %zprj.

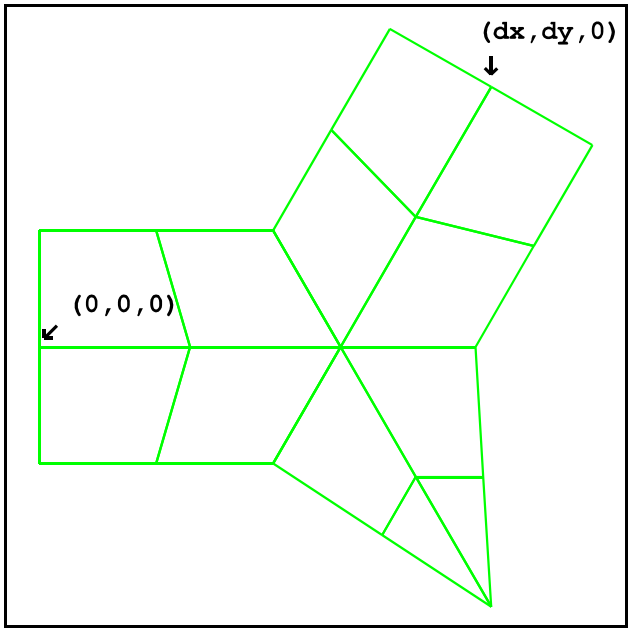
This feature can be used when building a sequence of parts that must start where the previous part ended. Each component can be constructed in a local coordinate and then transformed to the final position using a coordinate transformation. This command can be used to maintain a fiducial point as parts are added.

It is possible to do without this command by using only the replication commands. In some cases, calculating the translation vector can be complicated. This method simplifies the calculation of that translation vector.

Example

This example started with a single component by creating a command file named why.inc with the following commands:

```
block 1 3 0 4 5 6 0 8 10;
 1 2 3 0 5 7;
 1 2;
  [-2-1/sqrt(3)] [-1/sqrt(3)] 0
 -1 0 1 0
 [1/sqrt(3)] [2+1/sqrt(3)];
 -1 0 1 0 [1/sqrt(3)]
 [2+1/sqrt(3)];
 0 1;
dei 4 6; 1 3;;
dei 8 9; 5 6; 1 2;
dei 1 2; 5 6; 1 2;
```



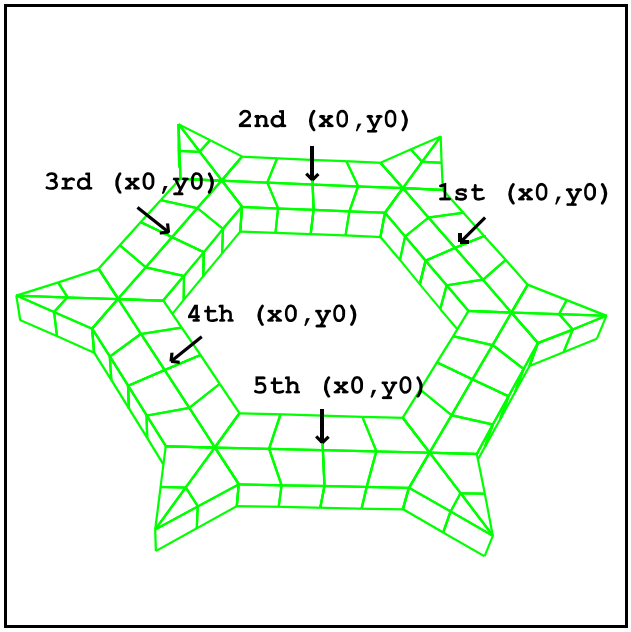
Component starting at the origin

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
tr 8 1 1 9 3 2 rz -60;
tr 4 5 1 6 6 2 rz -30;
pb 9 1 1 9 3 2
 xy 1.288675 -2.232051
pb 2 2 1 2 2 2 xy 0 0
pb 5 5 1 5 5 2 xy 0 0
pb 8 2 1 8 2 2 xy 0 0
mb 1 1 1 9 6 2 x [2+1/sqrt(3)]
```

To complete this example, a second command file is formed with the commands below. The component part starts at the origin, to simplify the calculations. Only the x and y-coordinates are maintained, since the part is formed in a plane, again for simplification. The vector (dx,dy,0) is formed between the starting and ending positions of the replicated part. This vector is transformed, just like the component part, to maintain the translation vector for the next replication.

```
para dx [2+1/sqrt(3)+cos(60)*(2+1/sqrt(3))] c differential vector
     dy [sin(60)*(2+1/sqrt(3))];
include why.inc
endpart
para x0 %dx y0 %dy; c 1st fiducial
include why.inc
gct 1 rz 60 mx %x0 my %y0;
grep 1;
endpart
trapt %dx %dy 0 gct 1
para x0%xprj c 2nd fiducial
     y0%yprj;
include why.inc
gct 1 rz 120 mx %x0 my %y0;
grep 1;
endpart
trapt %dx %dy 0 gct 1
para x0%xprj c 3rd fiducial
     y0%yprj;
include why.inc
gct 1 rz 180 mx %x0 my %y0;
grep 1;
endpart
trapt %dx %dy 0 gct 1
para x0%xprj c 4th fiducial
     y0%yprj;
include why.inc
```



Components rotated and translated

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

```
gct 1 rz 240 mx %x0 my %y0;
grep 1;
endpart
trapt %dx %dy 0 gct 1
para x0%xprj c 5th fiducial
     y0%yprj;
include why.inc
gct 1 rz 300 mx %x0 my %y0;
grep 1;
endpart
merge
stp .001
```

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

IV. Output

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

1. Output File Format Commands

A full description of the output commands can be found in the **True*Grid*®** Output Manual.

comment pass a comment to the DYNA3D output file

comment *[option [id]] text*

where the *option* can be

no	for the beginning of the output file
mt	for a material
np	for the nodes
he	for the hexahedron elements
be	for the beam elements
se	for the shell elements
ts	for the thick shell elements
sp	for a slide planes
nh	for the node history
hh	for the hexahedron element history
bh	for the beam element history
sh	for the shell element history
th	for the thick shell element history
br	for the brode
lc	for a load curve
nl	for the nodal loads
pr	for the pressure
ve	for the velocity
sw	for a stone wall
nc	for the nodal constraints
in	for the initial conditions
si	for a sliding interface
ti	for the tied nodes with failure
ex	for extra nodes w/ rigid body
jt	for a joint
xa	for the x-acceleration
ya	for the y-acceleration
za	for the z-acceleration
xv	for the x-velocity
yv	for the y-velocity
zv	for the z-velocity
dt	for a detonator

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

sd	for the springs and dampers
nr	for the non-reflecting boundaries
md	for momentum deposition in solid elements
1d	for 1D slide line

where *id* can be a number to identify the number of the type object being commented on

Remarks

Up to 100 comments can be saved. Not all types have an id associated with it and if a number is not the first thing encountered after the type, then the text is assumed to be general and not applied to a specific object. The id can be a parameter of an expression in square brackets.

epb element print block for DYNA3D and LSDYNA

epb *set_name*

where

set_name	name of an element set

mof mesh output file name

mof [*path*]*file_name*

where

path	path to the file - optional
file_name	file name including the suffix

Remarks

This function changes the name of the mesh output file to be written. It does not change the name of a mesh output file that has already been written.

The same thing can be accomplished by placing the o= option on the execute line. Under WINDOWS, the o= on the execute line is only possible when you run from a Command Prompt window. If you are running the WINDOWS operating system and you use a Command Prompt window to run or if you are running UNIX/LINUX/OSX, then the home directory is the directory specified in the **TGControls** window. If you specify a path, it will be relative to the home directory.

When you double click on a file in WINDOWS whose name includes the tg suffix, the same rules

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

apply.

When you start **True*Grid*®** from the desk top in WINDOWS, the home directory is determined from the settings in the **TGControls** window. Then the output file named in this command will be found relative to the working directory found in **TGControls**.

Be sure to issue the **write** command in the Merge phase to actually create the desired mesh output file.

ndigits number of digits written for coordinates

ndigits *n*

where *n* is the number of digits

Remarks

The default is 7 and the minimum is 5. This applies only to the Ale3d and CFX output options.

npb node print block for DYNA3D and LSDYNA

npb *node_selection*

where *node_selection* can be one of the following:

n *node*	select a node
rt *x y z*	select the node nearest to a point in Cartesian coordinates
cy *rho theta z*	select the node nearest to a point in cylindrical coordinates
sp *rho theta phi*	select the node nearest to a point in spherical coordinates
nset *set_name*	select all nodes in a node set

save dump buffered data to the tsave file

save *(no arguments)*

Remarks

On many systems, data is buffered before it is written to a file on disk. This command forces the data in the buffer to be written to disk. This guarantees that if there is a failure, the data will be saved in

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

the tsave file. This is usually not needed, because if, in the unlikely event, **True*Grid*®** should crash, the tsave file is properly closed first. However, this command is most useful if you want to use the data dumped to the tsave file without terminating **True*Grid*®**. By issuing the save command, you are assured that the tsave file is up to date.

verbatim write verbatim to the output file

verbatim
text
endverbatim

The text can be any number of lines of text. Anything on the same line and following the verbatim command is ignored. This text is saved. When the write command is issued, this text is written to the output file if the output file format is keyword driven so that the text can be in any order. This is needed because text is always placed near the beginning of the output file. Some of the formats supported by this feature are **Abaqus**, Ansys, **LSdyna**, **Marc**, and **Nastran**.

abaqus ABAQUS output format

abaqus (no arguments)

ale3d ALE3D output format

ale3d *(no arguments)*

ansys ANSYS output format

ansys *(no arguments)*

autodyn AUTODYN-3D output format

autodyn (no arguments)

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

cf3d Convective Flow output format

cf3d (no arguments)

cfd-ace CFD-ACE output format

cfd-ace (no arguments)

cfx CFX output format

cfx (no arguments)

dyna3d DYNA3D output format

dyna3d (no arguments)

es3d ES3D output format

es3d (no arguments)

exodusii Exodus II output format

exodusii (no arguments)

fidap FIDAP output format

fidap (no arguments)

fluent FLUENT output format

fluent (no argument)

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

gemini GEMINI output format

gemini (no arguments)

gridgen3d GRIDGEN output option

gridgen (no arguments)

iri IRI output format

iri (no arguments)

lsdyna LS-DYNA output format

lsdyna *type*
where
type can be:

fixed	for the fixed format
keyword	for the keyword format

lsnike3d LS-NIKE3D output format

lsnike3d (no arguments)

marc MARC output format

marc (no arguments)

nastran NASTRAN output format

nastran (no arguments)

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

nike3d NIKE3D output format

nike3d (no arguments)
enike3d (no arguments)
nnike3d (no arguments)
fnike3d (no arguments)

nekton2d NEKTON2D output format

nekton2d (no arguments)

nekton3d NEKTON3D output format

nekton3d (no arguments)

ne/nastran NE/NASTRAN output format

ne/nastran (no arguments)

neutral Neutral output Format

neutral (no arguments)

refleqs REFLEQS output format

refleqs (no arguments)

starcd STARCD output format

starcd (no arguments)

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

plot3d PLOT3D output format

plot3d (no arguments)

poly3d Generic output format

poly3d (no arguments)

tascflow TASCflow output format

tascflow (no arguments)

topaz3d TOPAZ3D output format

topaz3d (no arguments)

topaz3d2 TOPAZ3D version 2000 output format

topaz3d2 (no arguments)

viewpoint VIEWPOINT output format

viewpoint (no arguments)

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

2. Analysis Options

For control parameters and other such input options which are specific to the simulation code, use the appropriate command from this section.

To issue one of these commands, use its dialogue box. For full details, see the **True*Grid*®** Output Manual. But for the sake of completeness, the command names are given below.

XYZ Scientific Applications is actively adding support of more simulation codes. We welcome your requests for more output formats!

abaqstep **ABAQUS analysis step**

ansyopts **ANSYS analysis option**

dynaopts **DYNA3D analysis options**

lsdyopts **LS-DYNA analysis and database options**

marcopts **MARC analysis options**

nastopts **NASTRAN analysis options**

nenstopt **NE/NASTRAN analysis options**

nekopts **2d and 3d NEKTON 2.85 analysis options**

nikeopts **NIKE3D analysis options**

lsnkopts **LS-NIKE3D analysis options**

tz3dopts **TOPAZ3D analysis options**

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

3. Material Definitions

True*Grid*® generates input files for any of a large number of simulation codes. Each code has its own set of material models and its own way to specify them. You can specify your material models in **True*Grid*®** using a **True*Grid*®** dialogue box. For detailed information about material specifications, refer to the manuals of the appropriate simulation code. It is not practical to reproduce those manuals here.

For the sake of completeness, the currently available material definition commands are listed in the **True*Grid*®** Output Manual. To use these commands, use **True*Grid*®**'s graphical user interface.

XYZ Scientific Applications is actively adding support of more simulation codes. We welcome your requests for even more simulation codes!

abaqmats	**ABAQUS materials**
ansymats	**ANSYS materials**
dynaeos	**DYNA3D equation of state**
dynamats	**DYNA3D materials**
fluemats	**FLUENT materials**
nastmats	**NASTRAN materials**
nenstmats	**NE/NASTRAN materials**
lsdymats	**LS-DYNA materials**

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

lsdythmt	**LS-DYNA thermal materials**
lsdyeos	**LS-DYNA3D equation of state**
nikemats	**NIKE3D materials**
lsnkmats	**LS-NIKE3D materials**
marcmats	**MARC materials**
patsmats	**PATRAN materials**
tz3dmats	**TOPAZ3D materials**

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

V. Appendix

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Cartesian coordinate system

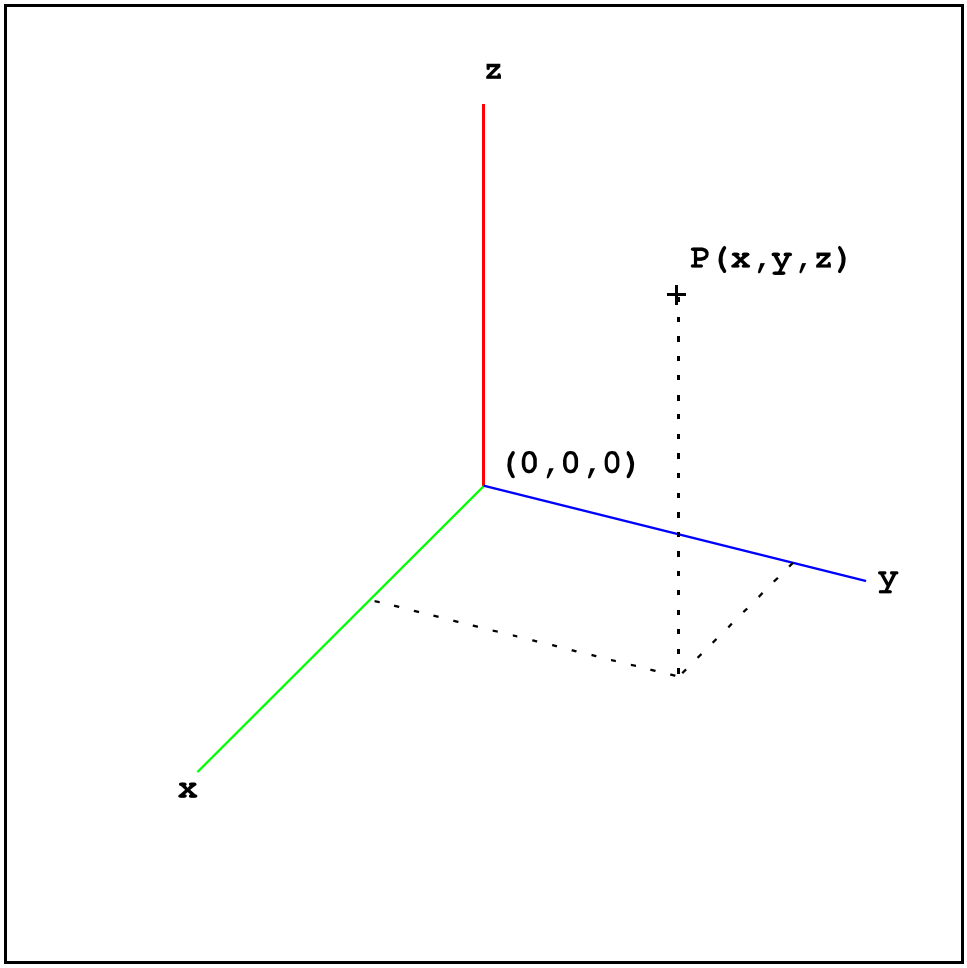


Figure 299 Cartesian coordinate system

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Cylindrical coordinate system

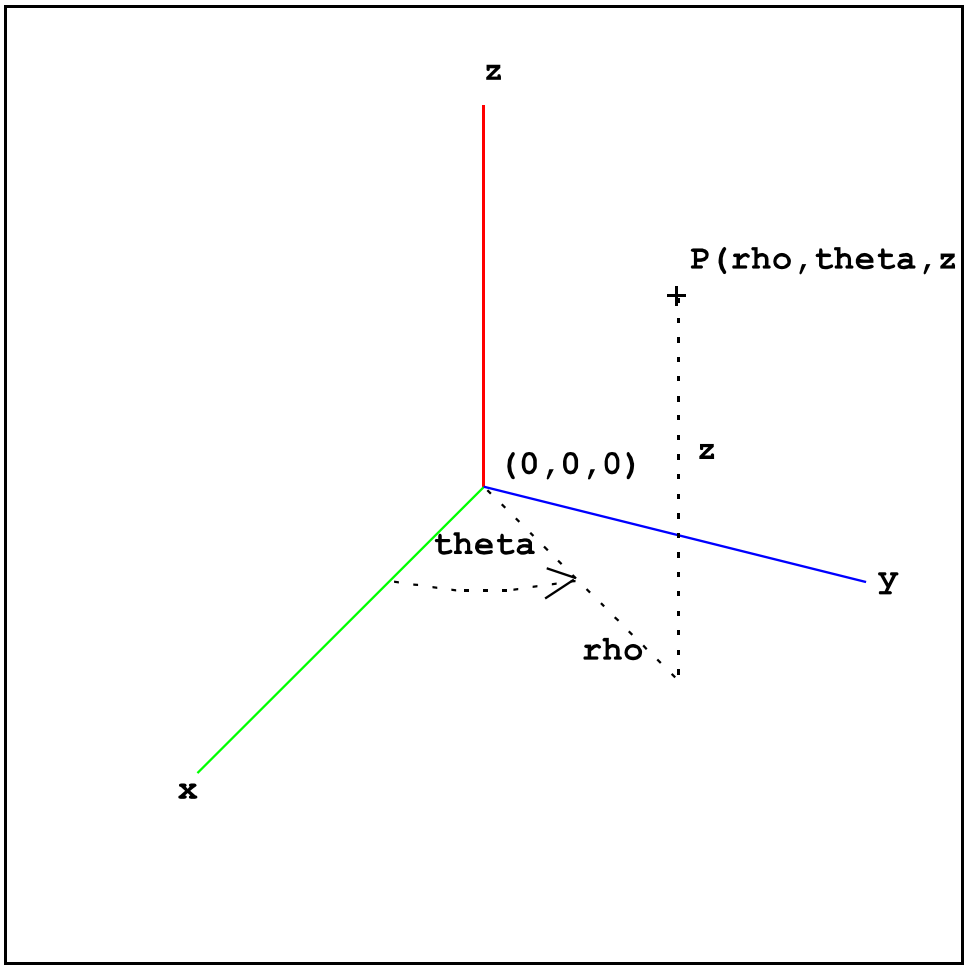


Figure 300 Cylindrical coordinate system

$$x = \cos(\theta)\rho$$

$$y = \sin(\theta)\rho$$

$$z = z$$

where ρ is radius, Θ is angle in x-y plane

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Spherical coordinate system

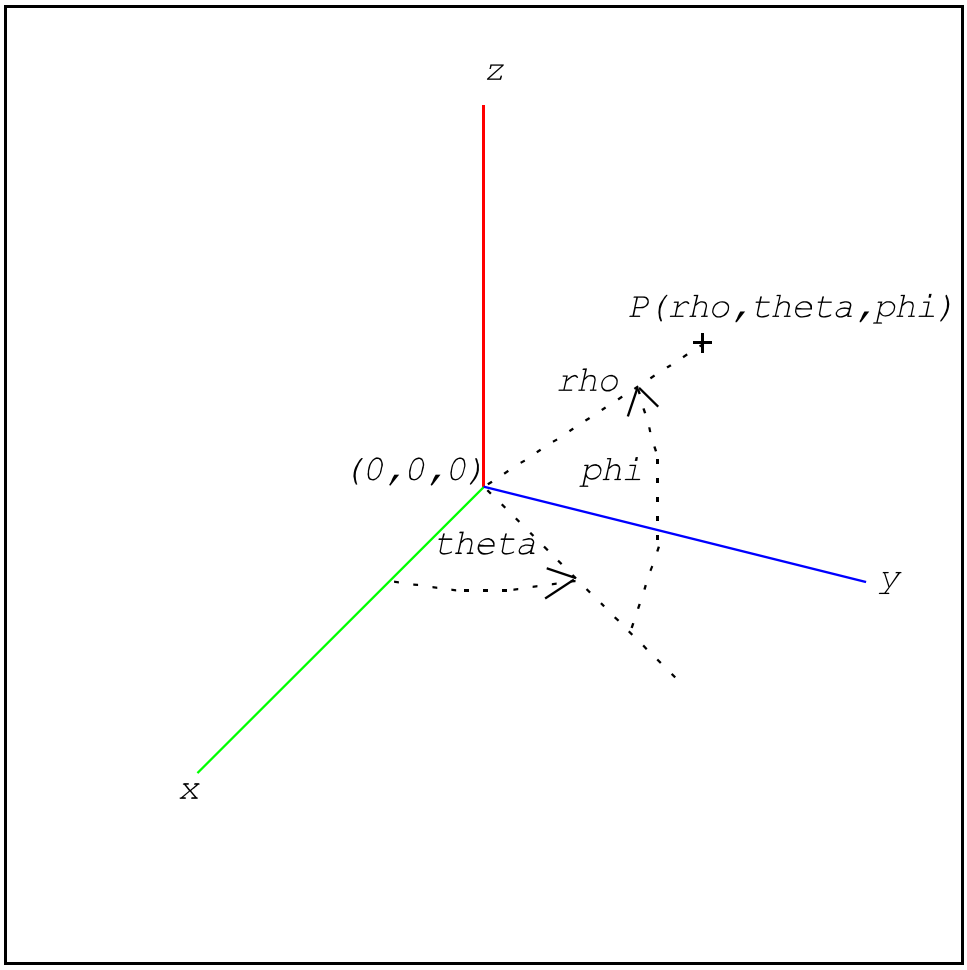


Figure 301 Spherical coordinate system

$$x = \cos(\theta)\cos(\phi)\rho$$

$$y = \sin(\theta)\cos(\phi)\rho$$

$$z = \sin(\phi)\rho$$

where
ρ is radius, Θ is angle in x-y plane, Φ is angle from x-y plane

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

VI. Index

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

.and.
expressions 427, 432
.eq.
expressions 427, 432
.eqv.
expressions 427, 432
.gt.
expressions 427, 432
.lt.
expressions 427, 432
.ne.
expressions 427, 432
.neqv.
expressions 427, 432
.not.
expressions 427, 432
.or.
expressions 427, 432
;
display commands 174
index list 344, 348
surface trans 113
!
expressions 427, 432
!=
expressions 427, 432
(...)
def 451
{..} 444
+
expressions 426, 431, 449
node set 330, 334
-
element set 325
expressions 426, 431, 449
face set 326
node set 330, 334
*
expressions 427, 431
**
expressions 427, 431
/
expressions 427, 431
^
expressions 427, 431
<
expressions 427, 432
<=
expressions 427, 432
=
element set 325
face set 326
node set 330, 334
==
expressions 427, 432
>
expressions 427, 432
>=
expressions 427, 432
%
def 451
para 455
&
expressions 427, 432
&&
expressions 427, 432
θ-coordinate 345
2D Curves 18, 36
3D Curves 25
closed 53, 56
composite 36
coordinates 27
Csp2 53
Ctbc 60
Ctbo 61
cubic spline 53
curvature 21
display 65
extruded/lofted 121
from a file 33
Ftbc 61
Ftbo 63

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Fws2 56
info 26
intersection 22
It 30
Lad 48
Lap 44
Lar 45
Lat 47
Lep 41
Lfil 44
Lint 52
Lnof 43
Lod 42
Lp2 36
Lpil 37, 38
Lpt 46
Lq 37
Lstl 49
Ltas 39
Ltbc 50
Ltbo 51
Ltp 46
Lvc 49
pan window 21
revolved surface 122, 124, 126
Rseg 64
ruled surface 161
segments 18, 19, 36
surface 106, 113
swept surface 167
to 3D Curve 68
volume definition, vd 112
window size 21
2nd order elements 341
3D Curves 66
2D Curves 25
3Dfunc 68, 86
accuracy 101
Acd 175
Acds 175
appending 69
Arc3 . . . . . 68, 89
attach 445
blend 3 surface 113
blend 4 surface 114
Bsp3 . . . . . 68, 79
closed 93
Contour 68, 74
Cpcd 68, 91
Cpcds 68, 92
Csp3 . . . . . 68, 75
csp3 example 114
Dacd 175
Dcd 175
Dcds 175
display 174
element set 325
Igc 68, 71, 184
IGES 181
Intcur 68, 84
labeled points 247
LD2D3D 68, 82
Lp3 . . . . . 68, 73
Lp3pt 68, 85
Nrb3 . . . . . 68, 80
nset 330
numbers 181
pipe surface 153
Point numbering 69, 70, 74
Projcur 68, 87
Pscur 68, 87
Racd 175
Rcd 175
Rcds 175
revoled surface 160
rmseg 68
rotation 160
ruled surface 123, 162
Sdedge 68, 71
Se 68, 71
segments 67
surface 107, 113

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Twsurf 68, 93
3Dfunc (Curd option) 68, 70, 86
Abaqmats material 473
Abaqstep
constraints 303
Abaqstep analysis option 472
Abaqus 467
beams 384, 408
load set number 303
offsets 407
verbatim 467
Abb 175
Abort
block 344, 348, 354
Abs expressions 450
Abs function 427, 432
Abscissa
2D Curves 18
Acc 293
display 215, 217
information 328
nset 330
remove 335
restore 337
Acc (Co option) 217
Accc
display 215, 217
Accci
display 215, 217
Acceleration
acc 293
condition display 215
frb 296
vacc 297
Acci
display 215, 217
Accs
display 215, 217
Accsi
display 215, 217
Accuracy 99, 182
example 243
getol 101
Acd 95, 175
Igesfiles 182
Acds 95, 175
Acos expressions 450
Acos function 428, 433
Add
lev 416
Adnset 322
example 323
Agrp 175, 198
Ajnp 205, 456
Ale3d 467
ndigits 466
Algebraic
3D Curve 68
surface 107
Algebraic Expression 447
Alv 175, 197
sd 109
Am 175, 268
Ams 175, 268
And
element set 325
face set 326
node set 330, 334
Angle
between lines 458
labels 246
Angles in expressions 449
Angular-coordinate 345, 349
Animation 263
Ansd 176
example 177
Ansymats material 473
Ansyopts analysis option 472
Ansys 467
beam cross sections 388
boundary conditions 317
cvt 317

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

offsets 407
verbatim 467
Ap 175, 359
Apld 19
segments 36
Apply button 174
Aps 175, 359
Arc
2D 20
2D fillet 44
2D, center and angle 48
2D, center and point 44
2D, radius and 2 points 45
2D, radius and tangent 46, 47
2D, radius, points, and tangent . . . 46
elliptic 41
Arc3 (Curd option) 68, 69, 89
Asd 175
Igesfile 182
sd 109
Asds 175
Asin expressions 450
Asin function 428, 432
Atan expressions 450
Atan function 428, 433
Atan2 expressions 450
Atan2 function 428, 433
Attach
3D curve 66
default 445
edge 66
Attach button
pvpn 105
Attaching
curtyp 445
Attch button 445
AUTODYN 467
Automatic contact 375
Av 263
Avc 263, 264
B 302
display 213
example 229
lb 307
nset 330
B-Spline
3D Curve 68
surface 116
Backplane 211
Bar
rigid, rbe 289
Batch execution 451, 453, 456
Batch file 442, 453
bb 456
Abb 175
Abbs 175
Dabb 175
data points 374
Dbb 175
Dbbs 175
display 214
errmod 449
example 220, 234, 244, 355, 372
intra-part 355
intra-part example 355
labeled 247
list 372
normal offset 355
Rabb 175
Rbb 175
Rbbs 175
retrieve 373
store 373
usage 441
Bbinfo 372
example 373
Beam 383
beam parts 356
Beams 356
ABAQUS 384
ANSYS 388
bm 383

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

cbeam 358
cross section, change 285
cross sections 383
display 246
DYNA3D 391
element offset 383
ibm, ibmi 383
integration rules 383
jbm, jbmi 383
kbm, kbmi 383
LS-DYNA 393
MARC 397
NASTRAN 400
NE/NASTRAN 405
part 341, 358
readmesh 350
scaled 383
Becho 443
Beep 443
Bf 316
display 215
Bf (Co option) 217
Bfd 369
display 215
Bfi
display 215
Bi
display 213
example 220
lb 307
Bind 275, 383, 406
Blend3 (Sd option) 107, 113
Blend4 (Sd option) 107, 114
Block 342
boundary display 174
boundary list 372
cylinder 349
example 222
part 341
partmode 292, 341
Block comments 444
Blude 354
faceset 138
part 341
partmode 292, 341
Bm 276
example 233, 246, 248
Npm 272
Bms 285
Bnstol 436
Body
rigid, rbe 290
Boundary conditions
algebraic surface 98
b 302
convection 316
convection thermal load 317
cubic spline surface 127
cubic surface 140
display 212, 213
flux 317
local 214, 307
si 270
Boundary radiation 318
display 213
Box
volume definition, vd 112
Bptol 201, 435, 440
bnstol 437
Npm 272
Break 433
Bricks
display 246
ordering 423
readmesh 350
Bsd 275, 383
bm 279
deform rods/bars 294
Bsinfo 405
lsbsd 407
Bsp3 (Curd option) 68, 79
Bsps (Sd option) 107, 116

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

boundary conditions 98
Bstl (Sd option) 108, 118, 182
features, fetol 100
mvpn, modify 102
pvpn, modify 105
Bulc 443
example 444
Bulk fluid 369
bf 316
bfd, properties 369
display 215
Buoy 267
Butterfly
3D Curves 84
Butterfly topology 443, 457
Bv 294
(Co option) 229
display 214
example 229
Bvi
display 214
C 444
CAD 179
acronym 196
Agrp 175
Alv 175
Dgrp 175
Dgrps 175
display 197-199
Dlv 175
Dlvs 175
importing geometry 179, 196
Rgrp 175
Rlv 175
Calculator 447
Caption 459
title 457, 459
Cartesian coordinate system 476
Cbeam 358, 383
part 341
Cdinfo 95
Cenref 205, 210
Center
triangle 457
Centroid 205
Cf3d 468
Cfd-ace 468
Cfx 468
glued interface and co grtol 240
glued interfaces 215
ndigits 466
CFX4
display 215
Chkfolds 99
Circent 444
Circle
2D arcs 20
3D Curve 68, 89
Circle center
circent 444
Cj
joint and Jd 363
Ckl 21
Cmplt
3D arc 89
Cn2p (Sd option) 106, 118
infinite 98
Co
acc 215, 217
angle 215
bf 215, 217
bv 214, 229
cv 213, 221
cvt 213, 221
detp 215
dx 213
dy 213
dz 213
efl 226
eft 214
epb 214, 231
fc 213, 216

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

fd 213, 217
ffc 215, 300
fl 213, 220
fmom 213, 217, 301
frb 215, 296
ft 214, 223
fv 214, 223
grtol 215, 240
il 214, 244
interfac 214, 234
iss 214, 225
jt 214, 246
Mdep 214, 242
mom 213, 241
n 215, 238
npb 214, 230
nr 214, 225
off 215
ol 214, 244
or 214, 227
pm 214, 233
pr 213, 217
rb 218
re 213, 219
resn 215, 239
rx 213
ry 213
rz 213
sc 215, 237
sfb 214, 227
si 213
size 215
sp 214, 232
spw 215, 245, 312, 313
spwf 215, 245, 314
sw 214, 224
sy 213, 243
syf 214, 228
tepro 215, 236
thic 214, 235
tm 213, 222
trp 215
ve 214, 225
vhg 214, 227
Coedg
Sdedge, Se 72
Coedge 179
Colon list 175, 341
Command Prompt
output 465
Comment 444
to output file 464
Compose
3D Curve 68
Composite
2D curves 36
edges 179
surface 109, 163
surface edges 72
surfaces 179
Condition 213
boundary 302
command 212
fc 216
fd 217
Infol 306
multiple, mlabs 212
readmesh 351
Conductance 376
Cone
surface 106, 118, 120
Cone (Sd option) 106, 120
Constraint
boundary 302
boundary, display 213
boundary, local 214, 307
Cont 350
Contact surface 375
Contact surfaces
condition display 213
Contour 67, 74
curd 70

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Point numbering 69, 70, 74
Contour (Curd option) 68, 74
surfaces 109
Control
phase 344, 348, 354
Control points
2D cubic spline 54, 57
B-Spline surface 116
cubic spline surface 127
cubic surface 140
NURBS surface 150
surface 98
Convection
boundary condition display 213
boundary conditions 316
cv 316
cvt 317
Convection thermal load
boundary conditions 317
display 213
Coordinate system
2D curves 18
Cartesian 476
Cylindrical 477
Local display 214
Spherical 478
surface 109
surface, local 113
coordinates
2 lists 37
2D Curve 27
labeled points 99
modify polar 51
pairs 36
polar 49-51
smallest 440
Copy
3D Curve 68
Cos expressions 450
Cos function 427, 432
Cosh expressions 450
Cosh function 428, 433
Cosine (Flcd option) 31
Cp (Sd option) 106, 121
infinite 98
Cpcd (Curd option) 68, 70, 91
Cpcds (Curd option) 68, 70, 92
Cr (Sd option) 18, 106, 122, 147
Cracks
display 247
Cross product 445
Cross section
beams 285
Crprod 445
example 445
Crule3d (Sd option) 123
Crvnset 324
example 324
Crx (Sd option) 18, 106, 124, 147
Cry (Sd option) 18, 106, 125, 147
Crz (Sd option) 18, 106, 126, 147
Csca 421
Curd 69
Gct 414
Iges 183
Igescd 186
Igespd 187
Igessd 189
Lct 412
lev 417
Nurbsd 190
Sd 108
Thickness 276, 383
vpsd 194
Csp2 (Ld option) 53
Csp3 (Curd option) 68, 70, 75
example 114
Csps (Sd option) 107, 127
boundary conditions 98
Ctbc
Ctbo 61
Ftbc 63

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Ctbc (Ld option) 60
Ctbo (Ld option) 61
Cubic spline
2D curve 53, 56
2D, polar 60
3D Curve 68
derivatives 77
polar 61
polar, modify 61
theory 76
Cur 67
attaching 445
curtyp 445
example 446
Curd 18, 68, 456
3Dfunc 86
Arc3 89
Bsp3 79
Contour 74
Cpcd 91
Cpcds 92
crule3d 123
Csp3 75
example 160, 248
Igc 71
IGES curve 184
Intcur 84
LD2D3D 82
Lp3 73
Lp3pt 85
Nrb3 80
Projcur 87
Pscur 87
rule3d 162
Sdedge 71
Se 71
surface 113
Twsurf 93
vpsd 194
Cure 67
attaching 445
curtyp 445
example 446
Curf 67
attaching 445
curtyp 445, 446
Curs
attaching 445
curtyp 445
example 446
Curtyp 445
Curvature 66
surface 102
twsurf 93
Curve
2-d 36
display numbers 247
curve rotation
for surface 125
Cusp 66
Cv 316
display 213
Cv (Co option) 221
Cvi
display 213
Cvt 317
display 213
Cvt (Co option) 221
Cvtab 349
Cvti
display 213
example 221
Cy (Sd option) 106, 132
infinite 98
Cy3 (Sd option) 106, 133
Cycorsy 349, 353
cylinder
generalized surface 121
interpolation 349
part 341
part definition 345
part example 216, 226

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

partmode 292, 341
surface 106, 132-135
volume definition, vd 112
xcy surface 169
ycy surface 171
zcy surface 173
Cylindrical coordinate system 477
cylindrical coordinates 348
Cylindrical Joint 363
Cyr2 (Sd option) 106, 134
infinite 98
Cyr3 (Sd option) 106, 135
infinite 98
Dabb 175
Dacd 94, 175
Igesfiles 182
Dam 175, 269
Dampers
properties 369
readmesh 350
Dap 175, 359
Dasd 175, 178
composite surfaces 163
Igesfile 182
sd 109
Data
experimental, surface 149
Dbb 175
Dbbs 175
Dc 447
example 447, 451, 452
Dcd 94, 175
Igesfiles 182
Dcds 94, 175
Igesfiles 182
De
example 221
Def 451
example 451
Default interpolation 454
Defeaturing 179
Deform rods/bars 294
Dei
example 218, 220, 225, 243
Delcd 96
Delcds 97
Delem 287
readmesh 353
Delete
element set 408
face set 408
Materials 341
node set 408
Delmats 341
Delsd 99
Delsds 100
Delset 408
Delspds 275
Derivatives
2D cubic spline 53, 61
boundary conditions 98
cubic spline surface 127
cubic surface 140
natural 98
polar cubic spline 60
Desk calculator 447
Detonation 362
Detonation point
detp 362
display 215
Detp 362
display 215
Dgrp 175, 198
Dgrps 175, 199
Dialogue box
bsd 405
offset 408
sid 382
spd 371
Dis 294
Displacement
display 213

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

fd 295
frb 296
initial 294
Display
commands 174
conditions and labels 212
list button 174
saving and restoring 264
Distance 447
Dlv 175, 197
Dlvs 175, 197
sd 109
Dm 175, 269
Dms 175, 269
Dom
example 245
Dot product 452
Dp 175, 359
Dpic 211
rpic 212
Dps 175, 360
Dsd 175, 178
Igesfile 182
sd 109
Dsds 175, 178
sd 109
Dummy arguments
def 451
Dummy interface 382
Dyna3D 468
beams 391
readmesh 349
Dynaeos Equation of State 473
Dynain
detached elements 210
readmesh 349, 353
Dynamats material 473
Dynaopts
analysis option 472
Dynaopts analysis 472
Echo 443, 448
example 429
Edge
3D Curves 71
example 235
surface interior 100
Edge file
rseg 64
Edge numbers
display 247
Edgefile 33
rln 35
rlns 35
Edges
degenerate, surface 110
Efl 321
display 214
example 226
Efl (Co option) 226
Efli
display 214
Electric flux
condition display 214
Element
detached 210
display 246, 253
print block, display 214
print block, epb 465
set, labeled 247
Element quality
Aspect ratio 209
Avolume option 208
Jacobian option 209
Orthogonal option 209
Pointvolume option 208
Smallest option 209
Volume option 208
Element set
delete 408
modify 325
readmesh 351
Electric flux

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

efl 321
Ellipsoid surface 136
Elliptic arc 41
Elm 205
measure 209
Elmoff 206
measure 209
Else 430
example 429
Elseif 429
example 429
End 448
end-of-line
UNIX 182
WINTEL 182
Endif 431
example 242, 429
Endpart 349
block 344, 348, 354
cylinder 348
errmod 449
surface 137
Endwhile 431
Enike3d 470
Epb 465
display 214
example 231
Epb (Co option) 231
Equation
def 451
Er
intp 147
surface 136
Er (Sd option) 106, 136
Erosion
contact 379
Errmod 448
Error 451
interrupt 448, 453
Es3D 468
Eset 325
detached 210
example 231
Esetc 409
Esetinfo 409
Esm
example 244
Etd 288
Exch 423
gexch 422
Excude 349
Exodusii 468
exp 265, 266
expressions 450
Exp function 427, 432
Exploded views 265
Expoff 265
Exp 265
Expression 447
def 451
para 455
Expressions 449
example 242
Fortran, format 426
line continuation 449
Extrude
2D Curve 18, 121
F5 Key 105
F7 key
distance 447
mbb 374
F8 Key 85
Fa 299
information 328
nset 330
remove 335
restore 337
Face (Sd option) 107, 137
features, fetol 100
Face set
blude 354
delete 408

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

display 247
example 355
modify 326
readmesh 351
Faces
display 246
Faceset (Sd option) 107, 138
features, fetol 100
mvpn, modify 102
pvpn, modify 105
Fbc 304
Fc 299
display 213, 216
example 216, 336
information 328
nset 330
remove 335
restore 337
Fc (Co option) 216
Fcc
display 216
Fcci
display 216
Fci
display 213, 216
Fcs
display 216
Fcsi
display 216
Fd 295
cylinder 349
display 213, 217
information 328
nset 330
remove 335
restore 337
Fd (Co option) 217
Fdc
display 213
Fdci
display 213
Fdi
display 213, 217
example 220
Fds
display 213
Fdsi
display 213
Features 100
Fetol 100
bstl 118
bstl example 118
face 137
Faceset 138
poly 158
polygon surfaces 107
stl 166
stl example 166
Ffc 300
display 215
Fidap 468
File
name, output 465
Finit (Flcd option) 32
Fl 317
display 213
Fl (Co option) 220
Flcd 31
Fli
display 213
example 220
Fluent 468
FLUENT material 473
Flux
boundary condition display 213
boundary conditions 317
fl 317
Fmom 300
display 213
Fmom (Co option) 217
Fnike3D 470
Foff (Flcd option) 32

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Follower
force 300
force, display 215
moment 300
moment, display 213
Force
display 216
FORTRAN
expressions 426
if statement 426
FORTRAN interpreter 449
Fowler-Wilson cubic spline 56, 61
Frame 410
Frb 296
display 215
Free edges
display 247
Free faces
display 247
Friction 376
Fsca (Flcd option) 32
Fset 326, 354
example 258
surface 138
Fsetc 409
Fsetinfo 410
Ft 318
display 214
information 328
nset 330
remove 335
restore 337
Ft (Co option) 223
Ftbc
Ctbo 61
Ftbc 63
Ftbc (Ld option) 61
Ftbo (Ld option) 63
Ftf
Curd 69
Gct 414
Iges 183
Igescd 185
Igespd 187
Igessd 188
Lct 412
lev 416
Lsys 366
Nurbsd 190
Sd 108
Trsd 110
vpsd 194
Fti
display 214
Function 451
Function (Sd option) 107, 139
Fv 295
display 214
example 223
information 328
nset 330
remove 335
restore 337
Fv (Co option) 223
Fvc
display 214
Fvci
display 214
Fvi
display 214
Fvs
display 214
Fvsi
display 214
Fvv 297
display 214
Fvvc
display 214
Fvvci
display 214
Fvvi
display 214

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Fvvs
display 214
Fvvsi
display 214
Fws2 (Ld option) 56
Gaps
between surfaces 163
Gaps in geometry 179
Gct 414
example 217, 242, 415, 426, 461
lev 417
trapt 460
Gemini 469
Generatrix
revoled surface 160
Geometry
accuracy 182
gaps 179
solids 179
trimmed surfaces 179
Geometry of the mesh 179
getbb 373
Getol 101, 182
revolved surface 160
trimming 192
Gexch 422
Global coordinate system 410
example 415
Gluing
display 215
st 438
stp 439
t 439
tp 440
Gmi 424
example 424
Graphics Commands
ignored 442
Gred
lev 416
Grep 415
example 461
lev 417
Npm 272
Pm 272
pslv 419
grid 98, 243
Gridgen3D 469
Group
boundary display 174
Grtol
condition display 215
thic 240
Grtol (Co option) 240
Gset 21
Gsii 425
example 425
Heat generation
display 214
vvhg 320
Hermite (Sd option) 107, 140
boundary conditions 98
History
File 451
interrupt 453
Home directory 465
I-coordinate 342
I-partition 342
Ibm 275, 383
Ibmi 275
element commands 383
Ibzone 442
If 426
example 242, 429
Igc 71
Igc (Curd option) 68, 71
Iges 67, 179, 183
3D Curve 68
accuracy 182
acronym 196
binary 180
binary data 180

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

composite curve 181
composite surfaces 163
conversion 182
entity dependencies 180
faults 180
file name 191
group 196
Igescd 186
Igespd 188
Igessd 189
importing geometry 179, 196
level 196
Nurbs 152, 181
nurbsd 190
other surfaces 181
parametric surfaces 143
plane 181
plane surfaces 145
readmesh 349
sd 107, 109
surface 97, 107, 151
surface sequence number . . 144, 145
surfaces 143
trimmed surface 181
useiges 193
utility 180
IGES surfaces 143
Igescd 67, 185
igesfile 184
trimming 192
Igesfile 184
Igescd 186
Igespd 188
Igessd 189
Nurbs 143, 145, 152
Nurbsd 190
Sd 107
useiges 193
Igesfind 180
utility 192
Igeslbls 186
Igesp (Sd option) 107, 145
Igespd 187
igesfile 184
Igessd 189, 190
numbering 181
sd 107, 109
surface 97
Igess
surface 143
Igess (Sd option) 107, 143
Igessd 188
igesfile 184
numbering 181
sd 107, 109
surface 97
Il 305
display 214
example 244
Il (Co option) 244
Ili
display 214
Importing geometry 179
Include 451
example 242, 452, 461
Indices
display 246
list 342
negative 342, 345, 348
reduced, display 246
zero 342, 345, 348
Inertia, moment of
reference point 210
infinite surface 71, 85, 119, 132, 148, 155, 156, 169-172, 174
Info 206
Infol 305, 327
example 336, 338
INGRID
beam 341
Iniexp 266
Infinite surface 98, 120, 157, 159, 173

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Initial Coordinates
block 345
cylinder 348
Initial Velocity 362
Inlet
display 214
il 305
Inner product 452
Inprod 452
example 452
Insprt
attach to curve 446
attachment 71
errmod 449
example 233, 429
Int 449
Int function 427, 432
Intcur 84
Intcur (Curd option) 68, 70, 84
Interactive execution 453
Interfac (Co option) 234
Interface
iss 270
save segments, display 214
Interface elements 375
display 214
Interpolate surface 147
Interpolation
2D curves 52
2D to 3D curve 82
3D curves 66
constraint, rbe 290
cylinder 349
default 454
mxp 459
Interrupt 453
errmod 448
resume 456
Intersection
2D curves 37
of surfaces 103
two surfaces 66, 68
Intp 147
Intp (Sd option) 105, 147
mvpn, modify 102
pvpn, modify 105
Inttr 374
example 374
Intyp 454
Inv
Curd 69
Gct 414
Iges 183
Igescd 185
Igespd 187
Igessd 189
Lct 412
lev 417
Lsys 366
Nurbsd 190
Sd 108
Trsd 110
vpsd 194
Iplan (Sd option) 106, 148
infinite 98
Iri 469
Iss 270
display 214
Iss (Co option) 225
Issi
display 214
Iterations 431
Itrim 191
J-coordinate 342
J-partition 342
Jbm 275, 383
example 232
Jbmi 275, 383
Jd 363
beam 358
display 214
example 246

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Joint 363
condition display 214
jt 306
merging 435
Jt 306
display 214
example 246
jd 363
Jt (Co option) 246
Jtinfo 365
K-coordinate 342
K-partition 342
Kbm 275, 383
Kbmi 275, 383
Key
F5 105
F7 447
F8 85
Labels 246
1D 248
2D 249
2q 249
3D 250
3q 250
command 212
cracks 262
crpvt 67
crpvt and Contour 74
crule3d 123
Crv 256
Crvpt 257
curd 71
facesel 247, 258
faceset 247, 258
fraces 261
fredges 261
ijk1 251
ijk2 251
ijk4 252
loc1d 253
loc2d 253
loc3d 254
loc3dq 255
locnd 252
multiple, mlabs 212
Nodes 248
nodeset 247
onset 247
rule3d 162
Sd 255
Sdedge 256
Sdpt 85, 257
sdpt and Contour 74
tol 260
vpsd 194
Lacd 95
Lad (Ld option) 48
Lap (Ld option) 44
Lar (Ld option) 45
Lasd 178
ansd 176
composite surfaces 163
IGES 197
Lasso
faceset 138
Last
Gct 414
Lct 413
lev 417
Lat (Ld option) 47
Lb 307
information 328
lsys 365
nset 330
remove 335
restore 337
Lcc 20
Lcd 22, 456
Lcinfo 25
Lcsd 101
Lct 412
example 207, 224, 233, 242,

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

245, 413, 426, 452
lev 417
trapt 460
ld 456
2D Curves 18
apld 19
Csp2 53
Ctbc 60
Ctbo 61
example 235
Ftbc 61
Ftbo 63
Fws2 56
Lad 48
Lap 44
Lar 45
Lat 47
Lep 41
Lfil 44, 226
Lint 52
Lnof 43
Lod 42
lp2 . . . . . 19, 36
Lpil 37
Lpt 46
Lpta 38
Lq 37
Lstl 49
lt 30
Ltas 39
Ltbc 50
Ltbo 51
Ltp 46
Lvc 49
revolved surface 124-126
Rseg 64
rule2d 161
segments 36
surface 113
Ld2d3d (Curd option) 68, 70, 82
Ld3d2d 25, 67
Ldinfo 26
Ldprnt 27
Lep (Ld option) 41
Lev 416
example 217, 418
pslv 419
Level 416
boundary display 174
scope 419
Lfil (Ld option) 44
Limits
bptol 441
para 455
ptol 441
Lin 67
example 454
intyp 454
Linear 291
elements 291, 341
example 248, 250
Lint (Ld option) 52
List 175
Ll
display 213
Lmi 425
example 245, 424
Lnof (Ld option) 43
Load
curve, definition 22
curve, display 65
curve, flcd 31
curve, info 25
curve, readmesh 350
curves, set id 23
display 213
nodal, fc 299
Load
curve 18
Load case
readmesh 350
Loads

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

time dependent 18
Local coordinate system 410
display 214
example 413
lb 366
Lod (Ld option) 42
Loft
2D curve 121
Log expressions 450
Log function 427, 432
Log10 expressions 450
Log10 function 427, 432
Logarithms expressions 450
Logical operators 427, 432
Loop 431
2D Curve 53, 56
Lp (Flcd option) 31
Lp2 (Ld option) 36
example 126
Lp3 (Curd option) 68, 70, 73
Lp3pt (Curd option) 68, 70, 85
Lpil (Ld option) 37
mazt 22
Lpt (Ld option) 46
Lpta (Ld option) 38
Lq (Ld option) 37
lrep
example 207, 224, 233, 245
lct 413
lev 417
Npm 272
Pm 272
pslv 419
Lrl 27
Lrot 28
LS-DYNA
beams 393
LS-DYNA thermal material 474
LS-DYNA3D
readmesh 349
Lsbsb 407
Lsca 29
Lscx 29
Lscz 30
Lsdsi 376
Lsdyeos Equation of State 474
Lsdymats 473
Lsdyna 469
offsets 407
verbatim 467
Lsii 426
example 425
Lsnike 469
Lsnkmats material 474
Lsnkopts analysis option 472
Lstl (Ld option) 49
Lsys 365
lb 307, 308
Lsysinfo 366
Ltas (Ld option) 39
Ltbc
ctbo 61
ftbc 63
Ltbc (Ld option) 50
Ltbo 51
Ltbo (Ld option) 51
Ltbc 50
Ltp (Ld option) 46
Lv 65
sd 109
Lvc 65
Lvc (Ld option) 49
Lvi 65
Lvs 66
Magnetic flux
boundary conditions 318, 319
Maplabel 350
Marc 469
beams 397
offsets 407
verbatim 467
Marc analysis 472

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Marcmats material 474
Mass 207
Mate
example 220, 232, 245
readmesh 350
Material 456
Am 175
Ams 175
coordinate system, display 214
Dam 175
display 174
Dm 175
Dms 175
element set 325
nset 330
Ram 175
Rm 175
Rms 175
Material number 350
increment 424, 425
Max expressions 450
Max function 427, 432
Mazt 22
lpil 37
Mb
example 222
Mbb 374
example 375
Mdep
display 214
example 242
Mdep (Co option) 242
Measure 207
Merge 437
across node sets 436
ambiguity 436
dummy interface 382
nodes 434
Parts 179, 247
phase 344, 348, 354
table 435
Mesh
intp 147
surface 149
Mesh (Sd option) 107
features, fetol 100
Mexp 266
Pexp 266
Min expressions 450
Min function 427, 432
mlabs
(Co option) 212
Mnl 203
Mns
example 438
Mod expressions 450
Mod function 427, 432
Mof 465
Mom 301
display 213
example 241, 336
infomation 328
nset 330
remove 335
restore 337
Mom (Co option) 241
Moments
display 213
mom 301
reference point 210
Momentum deposition
display 214
Momi
display 213
Move Pts. button
pvpn 105
Mp 321
infomation 328
nset 330
remove 335
restore 337
Mpc 308, 363

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Ms
sd 109
Mseq
example 233
Mt
readmesh 350
Mti
example 220
Mtv
volume 112
Multiple Point Constraint 363
joint 363
Mvnset 328
example 329
Mvpn 101
polygon surface 107
Mx
Curd 68
Gct 414
Iges 183
Igescd 185
Igespd 187
Igessd 188
Lct 412
lev 416
Lsys 365
Nurbsd 189
Sd 108
Trsd 110
Vpsd 193
mxiridx 456
mxjridx 456
mxkridx 456
Mxp 459
My
Curd 68
Gct 414
Iges 183
Igescd 185
Igespd 187
Igessd 188
Lct 412
lev 416
Lsys 366
Nurbsd 189
Sd 108
Trsd 110
vpsd 193
Mz
Curd 68
Gct 414
Iges 183
Igescd 185
Igespd 187
Igessd 188
Lct 412
lev 416
Lsys 366
Nurbsd 189
Sd 108
Trsd 110
vpsd 193
n
condition display 215
display 215
example 238
N (Co option) 238
Nastmats material 473
NASTRAN 349, 469
beams 400
deform rods/bars 294
offsets 407
readmesh 351
rigid bodies, rbe 290
spc 303
verbatim 467
NASTRAN analysis option
analysis option 472
Ndcons 350
Ndigits 466
NE/NASTRAN
beams 405

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

deform rods/bars 294
rigid bodies, rbe 290
spc 303
NE/NASTRAN analysis option
analysis option 472
NE/NSTRAN
output 470
Nekopts analysis option 472
Nekton2D 470
Nekton3D 470
Nenstmats material 473
Nerl 423
Nesting
included files 451
Neutral 470
offsets 407
readmesh 349
Nextbb 456
Nextcrv 456
Nextlc 456
Nextln 456
Nextmat 456
Nike3d 470
Nikemats material 474
Nikeopts analysis option 472
Nint expressions 450
Nint function 427, 432
Nnike3d 470
Nodal loads
display 216
Nodal rotation
display 215
frb 296
Node 456
constraint 350
display 246, 252
ordering 423
print block, display 214
print block, npb 466
rotation, frb 296
Node set
add 323
delete 408
display 247
loads 327
modify 328
onset 333
ordered 324, 329
readmesh 351
rml 335
Nogui 442
Nonreflecting boundaries
condition display 214
nr 308
Noplot 442
Norm expressions 450
Norm function 428, 433
Normal
circle, circent 444
Normal offset
2D curve 42, 43
bb example 355
csps 128
Csps, example 129
hermite 140
nrbs 150
polygon set 334
Normal to surface 103
Normal vectors
display 215
Npb 466
display 214
example 230
information 328
nset 330
remove 335
restore 337
Npb (Co option) 230
Npll 102
npm 271, 369
display 214
example 207, 233, 273

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Pm 272
Spring 274
Nr 308
display 214
Nr (Co option) 225
Nrb3 (Curd option) 68, 70, 80
Nrbs (Sd option) 107, 150
boundary conditions 98
Nri
display 214
example 225
Nset 330
example 224, 230, 324, 329, 331, 332, 438
Mpc 308
onset 333
remove subset 339
rsl 337
usage 438
Nsetc 410
nseti
example 323
Nsetinfo 410
Numbered surface
sd 109
Numerical
data, surface 149
NURBS
3D Curve 68
sd 109
surface 97, 151
Nurbs (Sd option) 107, 151
NURBS surfaces 189
Nurbsd 189
igesfile 184
Igessd 189
numbering 181
nurbs 152
Sd 107
Off
labels 246
Offset 407
spw 313
Ol 308
display 214
example 244
Ol (Co option) 244
Oli
display 214
Onset 333
Onset (Labels option)
example 323
Or
display 214
element set 325
face set 326
node set 330, 334
Or (Co option) 227
Ordering
bricks 423
nodes 423
Ordinate
2D Curves 18
Orpt
bulc 443
cvt 317
example 218-221, 238, 444
Pr 301
Rb 318
re 319
usage 438
orthogonal
example 355
Outlet
display 214
ol 308
Output
file name 465
Overlapping
surfaces 163
Painfo 454
para 456

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Para
example 429, 447, 452, 461
paraboloid
surface 159
Parameter
example 429
Parameterization
project 103
surface 97
Parameters 292
example 242
getting information 454
painfo 454
predefined 456, 457
Parametric
block 345
geometry 179
hermite surface 140
Parametric 3D Curve 86
Parametric surface 139
Parenthesis 427, 432
def 451
Part
Ap 175
Aps 175
beam 341
block 341
blude 341
bm 341
cbeam 341
cylinder 341
Dap 175
display 174
Dp 175
Dps 175
number, display 247
phase 344, 348, 354
Rap 175
readmesh 341
Rp 175
Rps 175
Partition
cylinder 348
Partitions 342
Partmode 291, 341, 345, 348
Patch
example 234
Patran 470
PATRAN material 473
Pbs 67
Penetration 376
Periodic
hermite surface 142
Pessure
Pr 301
Pexp
exp 265
Mexp 265, 266
Phase (Flcd option) 32
Pi 456
Pick
faceset 138
Pick button 158
pvpn 105
Pinfo 360
Pipe (Sd option) 107, 153
Pj joint and Jd 363
Pl2 (Sd option) 106, 154
Pl3 (Sd option) 106, 155
infinite 98
Pl3o (Sd option) 106, 156
infinite 98
Plan
listing 368
symmetry constraint 302
Plan (Sd option) 105, 157
infinite 98
Planar Joint 363
Plane 367
display 213, 214
example 228, 243
IGES surface 145

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

listing 368
surface 105, 148, 155, 157, 170, 172, 174
surface, pl2 154
surface, pl3o 156
Sw 315
Plate
rigid, rbe 289
Plinfo 368
Plot 442
Plot3D 471
Pm 272, 273, 369
display 214
example 233
Npm 272
Spring 274
Pm (Co option) 233
Pmass 210
Pminfo 273
Pn 204
Point
3D Curve 69, 74
label, surface 97, 101, 104
Point mass 369
display 214
info 273
npm 271
pm 272
readmesh 350
Point numbering
3D Curve 70
Contour 69, 70, 74
Surface 69, 70, 74
Poly (Sd option) 108, 158, 334
features, fetol 100
mvpn, modify 102
pvpn, modify 105
Poly Surface button
pvpn 105
Poly3D 471
Polygon
surface, poly 158
surfaces, features 100
Polygon sets
create 334
Polygonal
3D Curve 68
3D curves 85
surface 107
Polygonal surface
ViewPoint format 193
Postscript 263
Pplv
example 217, 418
lev 417
pslv 419
Pr 301
display 213
example 217
Pr (Co option) 217
Pr (Sd option) 106, 159
infinite 98
Precision
double 99
Prescribed
boundary 296
Pressure 301
condition display 213
Pri
display 213
Prod
lev 416
Projcur (Curd option) 68, 70, 87
project 103, 456
3D Curve 68, 87, 88
button 103
composite surfaces 163
entire surface 66
using 3D curves 66
Projection
accuracy 99, 182
button 93
coordinates 103

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

method 179
normal 103
Pscur (Curd option) 68, 87
Pset 334
surface, poly 158
Pslv 419, 420
example 217, 418
lev 417
Npm 272
Pm 272
pplv 420
Ptmass 350
Ptol 441
bnstol 437
Npm 272
Pvpn 104
polygon surface 107
Quadratic 291
elements 291, 341
example 248, 250
Quality 66
R-coordinate 345
r3dc (Sd option) 107, 160
example 161
Rabb 175
Racd 95, 175
Radial-coordinate 345
Radiation
boundary condition 318
boundary condition display 213
enclosure 319
enclosure condition display 213
enclosure surface display 215
Radius of circle
circent 444
Raixs
Gct 414
Lct 412
Ram 175, 269
Rand expressions 450
Rand function 428, 433
Random numbers 428, 433, 450
Rap 175, 360
Rasd 175, 179
sd 109
Raxis
Curd 68
Iges 183
Igescd 185
Igespd 187
Igessd 188
lev 416
Nurbsd 190
Sd 108
Trsd 110
vpsd 193
Rays 27
Rb 318
display 213
Rb (Co option) 218
Rbb 175
Rbbs 175
Rbe 288
Rbi
display 213
example 218
Rcd 95, 175
Rcds 95, 175
Re 319
display 213, 215
Re (Co option) 219
Readmesh 341, 349
block boundary 374
detached elements 210
Example 355
Materials 341
part type 291
Rml 335
Springs 275
Rebar
sliding 375
Reduced indices

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

display 246
Reference 210
Cenref 205
Reflect 410
Refleqs 470
tpara 457
Rei
display 213, 215
example 219
Relational operators 427, 432
Relaxi
example 224
Remove
3D Curve segment 68
Remove button
composite surfaces 163
Repe
Gct 414
joint 363
Lct 413
lev 417
Replications
global 415
joint 363
local 413
Res
example 454
Resn (Co option) 239
Rest button 174
infinite surfaces 109
restore 98, 174
sd 109
Resume 456
errmod 448
interrupt 453
Revolute Joint 363
Revolve
2D curve 122, 124-126
3D curve 160
Revolved
2D Curve 122, 124, 126
Rgrp 175, 199
Rgseg (Curd option) 68
Rigbm 441
Rigid 308
beam, readmesh 350
readmesh 350
Rigid body elements
rbe 288
Rj joint and Jd 363
Rln 35
edgefile 33, 34
Rlns 35
edgefile 33, 34
Rlv 175, 198
sd 109
Rm 175, 270
Rml 309, 335
example 336
Rsl 311
Rms 175, 270
Rmseg 96
Rod
rigid, rbe 288
Rotate 410
2D curve 28
2D Curves 18
Rotation 361
3D curve 160
motion 361
Velocity 298
Rounding in expressions 449
Rp 175, 360
Rpic 212
dpic 211
Rpl
element set 325
Rps 175, 361
Rsd 175, 179
Igesfile 182
sd 109
Rsds 175

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Rseg (Ld option) 33, 34, 64
Rsl 310, 337
example 338
Rule2d
intp 147
Rule2d (Sd option) 18, 106, 161
Rule3d
intp 147
Rule3d (Sd option) 107, 162
example 161
Ruled surface 123, 161, 162
2D curves 18
Rvnset 339
example 339
Rx
Curd 68
Gct 414
Iges 183
Igescd 185
Igespd 187
Igessd 188
Lct 412
lev 416
Lsys 366
Nurbsd 189
Sd 108
Trsd 110
vpsd 193
Rxy
Curd 68
Gct 414
Iges 183
Igescd 185
Igespd 187
Igessd 188
Lct 412
lev 416
Nurbsd 190
Sd 108
Trsd 110
vpsd 193
Rxz
Lct 412
Ry
Curd 68
Gct 414
Iges 183
Igescd 185
Igespd 187
Igessd 188
Lct 412
lev 416
Lsys 366
Nurbsd 189
Sd 108
Trsd 110
vpsd 193
Ryx
Iges 183
Igescd 185
Igessd 188
vpsd 193
Ryz
Curd 68
Gct 414
Igescd 185
Igespd 187
Igessd 188
lev 416
Nurbsd 190
Sd 108
Trsd 110
Rz
Curd 68
Gct 414
Iges 183
Igescd 185
Igespd 187
Igessd 188
Lct 412
lev 416
Lsys 366

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Nurbsd 190
Sd 108
Trsd 110
vpsd 193
Rzx
Curd 68
Gct 414
Iges 183
Igespd 187
Lct 412
lev 416
Nurbsd 190
Sd 108
Trsd 110
vpsd 193
Save
Gct 414
Lct 413
lev 417
tsave file 466
Save view 264
Saveiges 191
3D Curves 71
binary 180
Iges 184
useiges 192
savepart
getbb 373
Sc
display 215
Sc (Co option) 237
Scale 410, 421
2D Curve 29, 30
Sclexp 267
Scope
of level 419
Sd 18, 67, 105, 456
2D Curves 18
Blend3 113
Blend4 114
Bsps 116
Bstl 118, 182
Cn2p 118
Cone 120
Cp 121
Cr 122
Crule3d 123
Crx 124
Cry 125
Crz 126, 226
Csps 127
Cy 132
cy3 133
cyr2 134
cyr3 135
display surface numbers . . . 247, 255
dsds 178
Er 136
example 222, 234, 235, 243, 255
face 137
faceset 138
Function 139
function example 26
hermite 140
IGES NURBS 184
IGES plane 184
IGES surfaces 184
Igesfiles 182
Igesp 145
Igess 143
intp 147
Iplan 148
lasd 178
Mesh 149
Nrbs 150
Nurbs 151
Pipe 153
pl2 154
Pl3 155
Pl3o 156
Plan 157
poly 158

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

polygon sets 334
Pr 159
r3dc 160
Rule2d 161
Rule3d 162
sds 66, 163, 176, 179, 186, 197
Sp 165
Stl 166, 182
surface 97
Swept 167
Ts 168
volume 112
Xcy 169
Xyplan 170
Ycy 171
Yzplan 172
Zcy 173
Zxplan 174
Sdedge (Curd option) 68, 71
surfaces 109
Sdegde (Curd option)
example 26
Sdinfo 112
sdint
contour 74
Sds
sd 66
Sds (Sd option) 105, 163, 179, 197
ansd 176
lasd 178
name 109
Se (Curd option) 68, 71
Segments
2D curves 36
Seqnc
3D arc 89
Session file 451
interrupt 453
Set ID
load curves 23
Set identification 303
Sets button
polygons 334
surface, poly 158
Sf 109
cylinder 349
IGES surfaces 182
project 103
sd 109
surface 97
Sfb
display 214
Sfb (Co option) 227
Sfi 109
cylinder 349
example 218, 222, 243, 454
project 103
surface 97
Shell
normals 215
Shells
display 246
element offset 383
integration rules 383
readmesh 350
scaled 383
Ship example 355
Shv 263, 264
Si 270
display 213, 214
information 328
nset 330
remove 335
restore 337
sid 382
usage 438
Sid 375
display 213, 214
dummy type 437
example 234, 382
set identification, constraints . . . 303
usage 438

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

with si and sii 382
Sign expressions 450
Sign function 427, 432
Sii
display 213, 214
Siinfo 382
Sin expressions 450
Sin function 427, 432
Sind 408
Sine (Flcd option) 31
Sinh expressions 450
Sinh function 428, 433
Size 210
labels 246
Sj joint and Jd 363
Slide lines 375
Sliding interface 375
condition display 213
dummy interface 382
increment 425, 426
info 382
merging 435
si 270
table 435
Smags 210
readmesh 353
Smgap 110
Smoothing
3D Curve 88
Smoothing constraint
display 215
Solids 179
Sp
display 214
intp 147
Sp (Co option) 232
Sp (Sd option) 106
Spd 370
example 206
Spdp
display 214
example 207
spd 371
Sphere 165
surface 106, 165
volume definition, vd 112
Spherical coordinate system 478
Spherical Joint 363
Spinfo 372
Spline
B-Spline surface 116
cubic surface 140
NURBS surface 150
surface 151
surface, cubic 127
Spotweld 311
contact 377
display 215
joint 363
merging 435
spw 312
spwd, properties 313
spwf, LS-DYNA 314
Spring 273
delete 275
display 214
example 207, 232
merging 435
Npm 272
properties 370
spd 371
Springs
readmesh 350
Spw 312
display 215
example 245
spotweld 312
Spw (Co option) 245
Spwd 313
display 215
spotweld 312
spw 313

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Spwf 314
display 215
example 245
Spwf (Co option) 245
Sqrt
example 451
Sqrt expressions 450
Sqrt function 427, 432
Ssf
display 214
intp 147
sd 109
Ssfi
display 214
example 235
St 438
display 215
Stp 439
surface 168
Starcd output 470
Stereo lithography 182
ASCII file 166
binary file 118
Stl (Sd option) 108, 166, 182
features, fetol 100
mvpn, modify 102
pvpn, modify 105
Stone walls
condition display 214
definition 367
nodal selection 314
Stp 439
bnstol 437
display 215
dummy interface 382
example 224, 233, 436, 462
faceset 138
Npm 272
usage 438
with mpc 308
Subang 458
Supblk
display 215
example 240
Surface
accuracy 101
algebraic, function 139
Asd 175
Asds 175
blend, blend3 113
blend, blend4 114
boundary 66
composite 66, 101, 163, 176, 178, 186, 196
concatenated, sds 163
cone, cn2p 118
cone, cone 120
contour 68
control points 98
coordinate system 109
Crule3d 123
curvature 102
cylinder, cp 121
cylinder, cy 132
cylinder, xcy 169
cylinder, ycy 171
cylinder, zcy 173
cylindrical, crule3d 123
Dasd 175
delete 99, 100
dictionary 113
display 174
display numbers 247, 255
Dsd 175
Dsds 175
edge 68, 98
edge composite 66
edge to 3D curves 71
element set 325, 327
ellipsoid, er 136
extract edge 109
face 355
face set 326

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

faceset 138
fold 99
folding 102
Function 139
gaps 110
gaps between 163
IGES, iges 143
importing 179
infinite 98, 109
info 112
interpolated, intp 147
intersection 66, 103
label points 97
labeled points 247
list button 174
mesh 149
modify polygons 101
multiple 176, 178
name 109, 175, 186
neighboring 176
neighbors 176
normal 103
nset 330
numbered 109
numbers 181
NURBS, nurbs 151
overlaps 163
paraboloid, pr 159
part face, face 137
periodic 75
Pipe 153
plane, igesp 145
plane, iplan 148
plane, pl3 155
plane, pl3o 156
plane, plan 157
plane, xyplan 170
plane, yzplan 172
plane, zxplan 174
Point numbering 69, 70, 74
polygon 97
properties 97
Rasd 175
revolved, cr 122
revolved, crx 124
revolved, cry 125
revolved, crz 126
revolved, r3dc 160
Rsd 175
Rsds 175
ruled, rule2d 161
ruled, rule3d 162
Sds 163
sphere, sp 165
spline, csps 127
STL file, bstl 118
STL file, stl 166
swept, pipe 153
swept, swept 167
tangency 196
tangent 176
tessellation 97
torus, ts 168
transform, trsd 110
trimmed 97, 179
Surface button
composite surfaces 163
surfaces
multiple 163
Sv 263, 264
Sw 314
display 214
example 224
information 328
nset 330
plane definition 367
remove 335
restore 337
Sw (Co option) 224
Swap coordinates 422, 423
Swept
2D Curve 18

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Swept (Sd option) 107, 147, 167
Swi
display 214
plane definition 367
Sy (co option) 243
Syf 315
display 214
example 228
plane definition 367
Syf (Co option) 228
Syfi
display 214
plane definition 367
Symbolic selection 98
Symmetry plane with failure
condition display 214
definition 367
Symmetry planes
condition display 213
definition 367
T 439
display 215
dummy interface 382
Npm 272
Tp 440
with mpc 308
Tables
load curves 22
Tan expressions 450
Tan function 428, 432
Tangent
2D arc 46, 47
2D curve 38, 39
composite surfaces 164
surfaces 176
Tascflow output 471
Te 319, 369
information 328
nset 330
remove 335
restore 337
Tm 320
Tei 369
Tm 320
Temp 368
Tm 320
Temperature
constant 368
constant, te 319
initial, display 213
initial, tm 320
prescribed 214
prescribed, ft 318
profile, display 215
profile, tepro 319
Tepro 319
display 215
example 236
infomation 328
nset 330
remove 335
restore 337
Tepro (Co option) 236
Tessellation 97
Tetrahedron 291
Tf 67
Curd 68
example 454
Gct 414
Iges 183
Igescd 185
Igespd 187
Igessd 188
intyp 454
Lct 412
lev 416
Lsys 366
Nurbsd 190
Sd 108
Trsd 110
vpsd 193
Tfi

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

blend 3D curves 114, 115
example 222
TGControls 466
Th
display 214
Thi
display 214
Thic
display 214
Thic (Co option) 235
Thickness
Beam 276, 383
bm 276, 383
shells, display 214
Tied contact 375
Time dependent 18
Tinit (Flcd option) 32
Title 456
Tj joint and Jd 363
Tm 320
display 213
example 222
information 328
nset 330
remove 335
restore 337
Tm (Co option) 222
Tmass 211
Tmi
display 213
Tmm 267
Toff (Flcd option) 32
Tolerance
2D Curves intersection 22
bnstol 436
bptol 440
geometry 101
merging 434
ptol 441
st 438
stp 439
t 439
tp 440
ztol 440
TOPAZ3D
boundary conditions 316, 317
output 471
Rb 318
Re 319
TOPAZ3D material 474
Topaz3d2
output 471
Topology
surfaces 196
torus
surface 106, 168
Tp 440
bnstol 437
display 215
dummy interface 382
Npm 272
with mpc 308
Tpara 457
Tr
example 222, 234, 461
intp 147
Tracer particles
display 215
trp 315
Trans
surface 113
Transform
3D Curve 68
Transformation
point 460
Transformations 410
IGES 183
IGES curve 185
IGES NURBS 189
IGES plane 187
IGES surfaces 188
level 416

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

surface 113
ViewPoint format 193
Translate 410, 421, 422
2D Curve 30, 49
Translational Joint 363
Trapt 460
Trbb
errmod 449
example 374
interpolation parameter 374
Triangle
surface 97
Tricent 457
example 458
Trigonometric
functions 449
load curve 31
Trigonometric functions 427, 432
Trimmed
NURBS surface 152
Trimmed surfaces 179
trimming 191
curve 97
surface 181
Triple point 443, 457
Trp 315
display 215
Trsd
Sd 105
transform surface 110
Ts (Sd option) 106, 168
Tsave 451
2D Curve coordinates 27
2D Curves, info 26
3D contour 74
coedge 72
interrupt 453
lcinfo 25
Tsave file 72, 451
Coedg 72
Comments 444
include 451
save 467
Sdedge, Se 72
spotweld 312
Tsca (Flcd option) 32
Twsurf (Curd option) 68, 70, 93
Tz3dopts analysis option 472
Uj joint and Jd 363
Unifm
example 244
Union
surfaces 163
Universal Joint 363
UNIX
end-of-line 182
unresolved dependencies 459
Update 356
Useiges 192
3D curves 71
binary 180
Iges 184
saveiges 191
V 321
Curd 68
Gct 414
Iges 183
Igescd 185
Igespd 187
Igessd 188
information 328
Lct 412
lev 416
Lsys 366
nset 330
Nurbsd 189
remove 335
restore 337
Sd 108
Trsd 110
vpsd 193
Vacc 297

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

Vcv
display 213
Vcvi
display 213
Vd 112
Ve 298
display 214
example 225
information 328
motion 361
nset 330
remove 335
restore 337
Ve (Co option) 225
Vei
display 214
Velocity 362
boundary, bv 294
boundary, display 214
frb 296
initial, display 214
initial, ve 298
Motion 361
prescribed, display 214
prescribed, fv 295
prescribed, fvv 297
Ve 298
Velocity
display 214
Verbatim 467
Vft
display 214
Vfti
display 214
Vhg
display 214
Vhg (Co option) 227
Vhgi
display 214
View
saving and restoring 264
ViewPoint
importing geometry 182
output 471
surface 193
Volume 112
Vpsd 182, 193
example 194
features, fetol 100
fetol example 100
Sd 107
surface 97
write 196
Vtm
display 213
Vtmi
display 213
Vvhg 320
display 214
Vvhgi
display 214
Warning
avoiding 448
Warpage 208
Wedge 291
wedge part
example 355
While 431
Whole
3D arc 89
Wiges
hermite surface 140
Window
none 442
size 21
WINTEL
end-of-line 182
Wrsd 196
poly surface 158
polygon set 334
save modifications 102, 105
Sd 107

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved

X-coordinate 342
Xcy (Sd option) 106, 169
infinite 98
Xnrm 103, 456
Xoff 421
Xprj 103, 456
bulc 443
example 458
predefined 103
transformed point 460
tricent 457
Xsca 421
Curd 69
Gct 414
Iges 183
Igescd 186
Igespd 187
Igessd 189
Lct 412
lev 417
Nurbsd 190
Sd 108
Thickness 276, 383
vpsd 194
Xyplan (Sd option) 106, 170
infinite 98
Y-coordinate 342
Y=
example 245
Ycy (Sd option) 106, 171
infinite 98
Ynrm 103, 456
Yoff 422
Yprj 103, 456
bulc 443
example 458
transformed point 460
tricent 457
Ysca 421
Curd 69
Gct 414
Iges 183
Igescd 186
Igespd 187
Igessd 189
Lct 413
lev 417
Nurbsd 190
Sd 108
Thickness 276, 383
vpsd 194
Yzplan (Sd option) 106, 172
infinite 98
Z-coordinate 342, 345
Z=
example 216, 225, 232
Zcy (Sd option) 106, 173
infinite 98
Znrm 103, 456
Zoff 422
Zprj 103, 456
bulc 443
example 458
transformed point 460
tricent 457, 459
Zsca 421
Curd 69
Gct 414
Iges 184
Igescd 186
Igespd 187
Igessd 189
Lct 413
lev 417
Nurbsd 190
Sd 108
Thickness 276, 383
vpsd 194
Ztol 440
Zxplan (Sd option) 106, 174
infinite 98

Copyright © 1992-2006 by XYZ Scientific Applications, Inc. All Rights Reserved